Sonification of Spectroscopy Data

By Matthew Pietrucha
Worcester Polytechnic Institute

Submitted to the Faculty of the WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the Degree of Master of Science in Interactive Media and Game Development

Primary Advisor:
VJ Manzo, Associate Professor of Music

Readers:
Scott Barton, Associate Professor of Music
Charles Roberts, Assistant Professor of Computer Science

List of Figures

Abstract

Sonification is the process of mapping non-audio data to sound. The field has three key areas of research: (1) psychological research in perception and cognition, (2) the development of tools (SDKs, packages, hardware connectivity), and (3) sonification design and application. The goals of this research were twofold: (1) To provide insights to the development of sonification tools within the programming environment Max for use in further sonification/interdisciplinary research, as well as (2) provide a framework for a musical sonification system.

The sonification system discussed was developed to audify spectroscopy data, with the purpose of better understanding how multi-purpose systems can be modified to suit a particular need. Since all sonification systems may become context specific to the data they audify, we developed a system in the programming language Max that is both modular and responsive to the parameterization of data to create musical outcomes. The trends and phenomena of spectral data in the field of spectroscopy plot musically through the system and further enhanced by processes that associate descriptors of said data with compositional idioms such as rhythm, melody, and harmony.

This process was achieved in Max by creating a modular system that handles the importing and formatting of spectral data (or any data in an array format) to send that data to a variety of subprograms for sonification. Subprograms handle timing and duration, diatonic melody, harmony, and timbral aspects including synthesis and audio effects. These systems are accessible both at a high level for novice users, as well as within the Max environment for more nuanced modification to support further research.

Acknowledgements

Many thanks to my advisor VJ Manzo for all his support. I may have never attended graduate school without him suggesting it to me years ago. (I think I'm still partial to Queen II, VJ. It only got one track in Bohemian Rhapsody, still the underdog album..) Thank you to my readers Scott Barton and Charlie Roberts, whose advice and suggestions improved my research and writing. Thank you to my father, Steven Pietrucha, for all his love and support. Lastly, I dedicate this work to my grandparents Joseph and Edna Pietrucha. Their love and support has made all things possible.

Introduction

The purpose of this research was to provide insight into the development of sonification tools and programs within the Max programming environment to help meet the expanding needs of the sonification field at large. By developing a novel and musical application aligned with a specific data type, the research also aims to illustrate how it and other similar systems may be modified to make sonification study more accessible and approachable. Barrass (2012) argues that "sonification is a design practice in which effective solutions to the problem of communicating data via sound are achieved through an iterative, heuristic process." This iterative process creates a need for modular, flexible, and expandible systems that can meet the demand for specialized and context specific sonifications.

One key issue in designing sonification systems is its inherent reliance on an interdisciplinary methodology, which requires the collaboration of programmers, scientists, and musicians with music theory, composition, sound design, and synthesis knowledge. Vickers (2004) noted that "...for sonification the goal is to maximize the information transfer while minimizing the noise (a high signal-to-noise ratio). However, for composers, music that contains high information and low redundancy cannot be accommodated into musical schemata."(Snyder, 2001: 235)

Our expectations as listeners are embraced, delayed, or denied, and these expectations tie to the amount of information in a musical piece. As such, music with low information tends to confirm our expectations more often. For example, popular music tends to be less demanding to a listener's expectations than other genres. Electroacoustic music varies in the degree of difficulty in regards to how listeners unpack musical information. Sonification operates somewhere between these distinctions. Sonifications may be electroacoustic, and their reliance on data to produce sound can make them procedural in nature. However, sonifications also exist as tools for multimodal/periphery monitoring and thus usually have some functional purpose outside of aesthetic experience. Overall, the musical experience of a sonification impacts the experience of data and ties to each listener's experience with music as a whole.

While many simple data-to-MIDI mappings for sonification exist, developers struggle to create aesthetically pleasing systems that are also useful in conveying data phenomena. For example, the TwoTone Data Sonification app developed by Datavized Technologies is flexible with loading multiple data sets and mapping each set to a musical scale. However, it does not visualize the data for a multimodal experience displaying a column of values. It also relies entirely on producing a single melodic voice from a data column, with the only adjustable parameters being pitch, duration, volume, and an instrument (timbre).

Loading in multiple data sets serves as an interesting compositional aid, but if sonifications are to simultaneously function as a tool for analysis or an augmented/periphery experience, then combining multiple sets as independent voices may be problematic as it introduces more

information, or noise, that is competing perceptually with the same space. Asking a listener to unpack simultaneous melodic voices and make sense of them can be a difficult task based on how parameterizations execute. The study "Drawing By Ear: Interpreting Sonified Line Graphs" by Lorna M. Brown and Stephen A. Brewster (2003) suggests in its results that multiple streams of data are better recognized by a listener when they use the same timbre, despite the user preference for them to be different.

Another example of a data agnostic system, or a system which is multi-purpose in nature in regards to how it handles many different types of data, is the Data to Music API by Takahiko Tsuchiya and Jason Freeman (2015). Its design allows users to modify high level music parameters, avoiding lower level one-to-one parameters, and offers this functionality as a real time process. The crux of this approach is the theory that all sonifications "...may have musical effects on listeners, as our trained ears with daily exposure to music tend to naturally distinguish musical and non-musical sound relationships, such as harmony, rhythmic stability, or timbral balance" (Tsuchiya et al., 2015). We all approach musicing (performing, listening, rehearsing, or composing) informed by a variety of learned cultural contexts. These contexts inform our understanding of the relationships between data and music.

Watkins and Dyson (1985) demonstrated that melodies following the rules of Western tonal music are easier to learn, organize cognitively, and discriminate than control tone sequences of similar complexity, however this result extends primarily to Western listeners. This suggests that the cognitive organizational overhead associated with atonal systems makes them less well suited as carriers of program information (Vickers 2004). The approach to our application, *Sonify,* uses similar data scaling and filtering methods as the aforementioned applications like TwoTone, by creating common diatonic relationships between significant data changes. More unique to the Sonify platform is how insignificant data changes behave. Repetitive data is filtered to silence, resulting in unique rhythmizations and phrasing from a data set. Additionally, other accompaniment voices such as a bass voice or chords can link to the same phenomena through other parameterizations. As both user preference varies in regard to parameterization, and this preference may impact the experience of data, these features allow for research into other musical phenomena not commonly featured in these other applications.

We chose the Max programming environment to increase the accessibility of sonification development for musicians and creatives, as Max is widely adopted in the music community. Max's visual object oriented design promotes rapid prototyping. As a data flow language, it also naturally lends itself well to manipulating and controlling real time streams of data, which is efficient for sonification work. These affordances encourage modular design. When combined with functionality like Max's package manager, it may shorten sonification development time for other programmers in the future.

Background

Sonification is the process of mapping non-musical data to sound, usually with the intent of conveying aspects of the data that would be difficult to visually express, by enhancing periphery monitoring (Hunt & Neuhoff, 2011). Additionally, sonifications occur over a duration of time, while data visualizations may be static in nature. While composers have explored timbral possibilities for centuries, the manipulation of timbre was constrained to interactions with acoustic instruments or the voice. In the past century, the advent of recording and synthesis has led to new possibilities of sonic manipulation, which has led to new compositional techniques. As noted by Lodha (et. al) in the design of MUSE, a musical data sonification toolkit, the creation of musical and non-fatiguing sounds is important in experiencing sonifications over extended periods of time. Achieving this goal alongside dealing with the subjectivity of parameterization has called for flexible programs in the field.

There is debate on whether sonification belongs to the realm of music, art or science, or all three (Sinclair, 2012). The need to comprehend complex streams of data has increased the need for periphery monitoring of data. This need coincides with increased access to more powerful media technologies (Walker, 2010). Early sonification traces back to geiger counters and stethoscopes, which use non-speech sound to warn the user of harmful radiation, or help us hear how the rhythms of our internal organs might suggest illness (Walker et al., 2010). In the realm of music, in the late 1950's, composers such as Xenakis experimented with applying mathematical probability tables to music in his piece *Achorripsis*, which applied those probability tables to different musical parameters. Xenakis also represented the statistical mechanics of gases in his piece *Pithoprakta (1955-56)*, where each molecule moving through space ties to a string instrument and pitch. He created his own simulation of temperatures and pressures, and created his own graphs of these simulations to aid in the composition. In the 1960's, composer John Cage used a variety of data sources in his compositions, such as *Atlas Eclipticalis*, which utilized star charts (Childs, 2018). While these pieces share some techniques of parameterization, they can be more broadly defined as "data driven music" instead of sonification, as their goals differ from sonification in that the latter is often used to some analytic end, whether that is periphery monitoring of data or the primary tool for analyzing data behavior.

Current sonification systems take advantage of our ease of access to complex data sets through web APIs, scientific tools, and databases. Interactive sonification systems enable users to adjust sonification parameters directly, often in real time.

Sonification also has many other practical purposes, designed around periphery audio information. One of the first industrial sonifications was the ARKola system, which notifies workers of every process at a bottling facility. ARKola was developed and tested by William W. Gaver, Randall B. Smith, and Tim O'Shea (1991). Each of the nine machines in the facility communicated its state through an "auditory icon", a sound effect tied to the machine's current state. For instance, a heating machine made a blowtorch sound, and a bottling machine made a clunking sound. The results of this research concluded that "...audio cues can provide useful

information about processes and problems, and support the perceptual integration of a number of separate processes into one complex one. In addition, they can smooth the transition between division of labour and collaboration by providing a new dimension of reference. These results suggest that auditory icons can play a significant role in future multiprocessing and collaborative systems" (Gaver et.al 1991).

Another example is the ShareMon system developed by Jonathan Cohen at Apple in 1993. ShareMon used a variety of sound effects from Star Trek to notify users of background network tasks on a computer. Hermann, Drees, and Ritter created a weather "prototypes" system that took 24-hour data streams of weather patterns and sonified them as non-speech weather reports for radio. Engineers from the 50s through 70s would tune their radios to pick up the electrical noise from their computer processors, and used those sounds to help inform debugging the computer (Hunt & Neuhoff, 2011). These "earcons" are simple sonifications where a short musical motif conveys a system's state. Another common example is the tone a user hears when they receive an error message using the Windows operating system.

The most common approach to data sonification is parameter mapping, where a data dimension maps onto an auditory parameter such as duration, pitch, loudness, position, brightness, etc. Different variables can map to different parameters at the same time to produce a complex sound. The parameter-mapping approach has the following advantages: (1) ease of production – existing tools allow mappings to many auditory parameters; and (2) multivariate representation – many data dimensions can be listened to at the same time (Barass & Kramer, 1999).

Parameterization may use a direct mapping of hierarchical music structures such as pitch, volume, harmony, and duration, which are the primary structural factors in Western tonal music. It may also map timbral aspects such as signal processing effects, by which audio signals transform through spectral processes such as reverberation, echoes, equalization, filtering, or saturation. The common obstacle of parameterization is how parameters map, sometimes arbitrarily, to data. The resulting sound is often unpleasant and usually lacks any natural connection to the data represented (Barass & Kramer, 1999).

One expects that spectral data sounds different from demographic data or atmospheric pressure readings – models of sonification must be sensitive to the type of data they are representing. Indeed, the "mapping problem" is the most significant impediment to an otherwise flexible and potentially powerful means of representing information (Worrall, 2011). To promote widespread acceptance of sonification, systems should "provide easy-to-use tools and systems that allow non-experts to make their own sonifications tailored to their particular task, data, experience and expectations" (Barass, 2011).

The mapping problem itself arises from the many approaches to create data sonifications. The three data representations in sonification identified by Alberto de Campo (2007) are *Continuous, Discrete Point,* and *Model-Based* representation. These representations all have their own advantages and shortcomings. *Continuous* treats data as "quasi-analog continuous signals", often tied to movement in a single axis. *Discrete Point* creates events for every data point for a more

modular, flexible approach to sonification. *Model-Based* introduces a musical model that is informed by the data, which can be useful for implementing the domain knowledge, or data behavior, directly into the musical model itself. As a drawback, this can produce bias as the model curates the experience around that domain knowledge. Campo is careful not to compartmentalize these representations, and his "Data Sonification Design Space Map" has overlapping zones that encourage combinations of these approaches based on the number of dimensions in the data as well as the number of data points.

While sonification of spectra data has been previously explored, other systems focus on real time auditory monitoring for the visually impaired or as a periphery monitoring process combined with haptics while using hardware with force-based spectroscopy. The study for sonified infrared spectra involving the visually impaired uses a CSV to MIDI conversion process (Pereira et.al 2013) was also conducted at Worcester Polytechnic Institute by a group of undergraduate students, and similarly maps frequency to absorption rates from spectra, but maps the wavelength value of X to a timing parameter. The study notes that "...the full interpretation of an IR spectrum is difficult because of its inherent complexity. Fortunately, it is not necessary to fully interpret an IR spectrum to get useful structural information." Similarly concluded through the Sonify development process was to emphasize certain aspects of spectra to gain a broad analysis of the molecular composition by hearing bands of absorption activity in key wavelength regions.

In terms of pedagogical uses of sonifications, the effectiveness of sonifications are partially determined by how specialized the parameterization is in the context of the data it is representing. For example, sonifications of sea levels, earth surface temperatures, and atmospheric CO2 levels created by Mark Bellora were presented to undergraduate students to assess how they may enhance understandings of the data sets. The study focuses on educational settings and was conducted by Robert Pockalny (Ballora et.al, 2018). This study differed from informal STEM contexts such as a museum, in that the sonifications were part of a class module on climate change. The preliminary results from this study indicate that the addition of sound was useful in understanding the data sets overall. Interestingly, these sonfications were static compositions with no interactivity or ability for the listener to change parameters to suit their needs. There is certainly merit to controlled output in gauging the effectiveness of sonifications in pedagogical contexts. However, as parameterization is subjective, the ability to modify the sonification may lead to different understandings on its effectiveness as a tool.

As such, future applications in the field should explore combinations of "agnostic" models, or models which allow for specialized data parameterizations to convey one specific form of data. The Sonify application's design process reflects trying to balance these two approaches to meet the needs of chemical engineering data sonification through specialized processes, as well as allowing for modifications to the system through the changing of musical models. The goal of including a model based approach was to allow an expert to describe data phenomena, and without musical knowledge, tie that data phenomena to some musical phenomena. Additionally, the expert can create distinct musical differences to differentiate selected regions for playback. Lastly, models increase accessibility by hiding more complex sound design and compositional

processes to increase immediacy in interacting with data.

Method

Overview

The objective of this research was to design a system that facilitates sonification of spectroscopy data in the Max programming environment, which could then be validated by an expert in chemical engineering at WPI, Associate Professor of Chemical Engineering Michael Timko. The sonification application was iteratively tested and built in collaboration with Professor Timko to suit his needs as an educator. Validation involved considering the application's parameterization and design in a theoretical pedagogical context during the interview process. This methodology has three major components:

1. The development of a software-based sonification system
2. Implementation of that system within the context of chemical engineering
3. Validating the built-in tutorial as an effective mechanism for conveying the basic operation the system

Development took place over a two year period following an iterative practice-led research approach (Skains, 2018). Practice-led research is a common paradigm in the creative arts, where "...the observations and experiences of practical circumstances often lead to new research questions" (Brown & Sorensen, 2009). In this model, a feedback loop often exists between speculation and experimentation. This led to iterations on the design of the application, as different models and processes were tested after each successive interview.

The original prototype focused on a multi-purpose sonification system, which was then modified specifically for spectroscopy data in its second phase of development. Alongside these specializations are the development of tools for the sonification system itself in Max, which required the creation of modules to process/filter data, interact with the data, and convert that data to audio signals through audio synthesis processes. Development had two guided factors:

1. Sonify must work as a standalone application by not requiring any additional software to run.

2. Both the sonification behavior and application user interface should allow someone knowledgeable about the data behavior to easily describe that data in musical terms via a tagging system. The tagging system imposes a musical model on the data sonification to occur without interfering with other parameterizations that fundamentally describe spectra phenomena.

The modifications in the second year were developed with Michael Timko, Associate Professor of Chemical Engineering. Professor Timko's insights and recommendations came from a combination of interviews used to optimize the system for spectral data. Notes were taken at

each interview stage to modify the system. Upon completion of development, we conducted a user study with undergraduate students to test the accessibility of the application before comprehensive research could occur in pedagogical contexts.

1. Software Development

Prototype

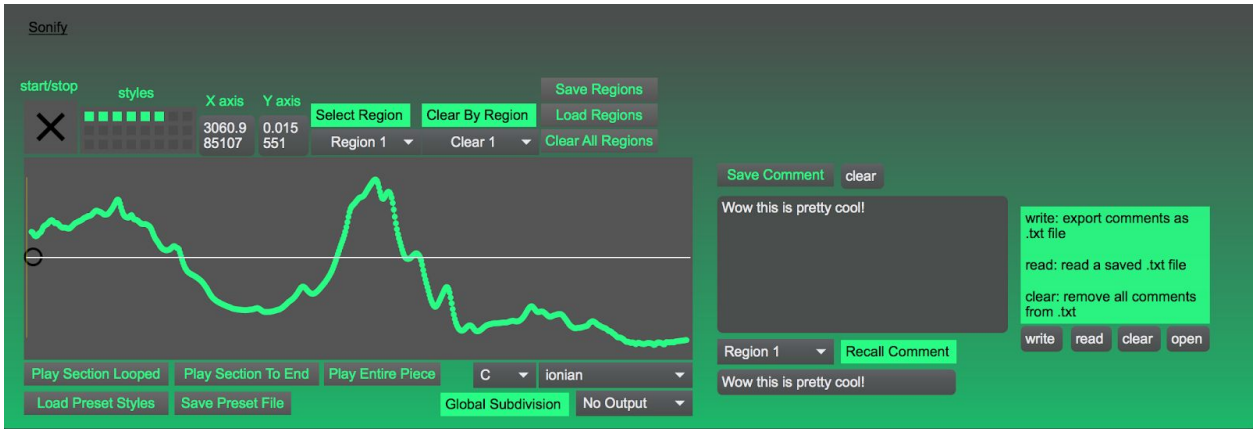


Figure 1: Year one prototype of sonification platform

Initial focus

The alpha build produced the basic functions necessary for a sonification system inside Max. The key features for this build were as follows:

1. Import any array data to be visualized in graph form. Find minimum and maximum values of X and Y for pitch scaling.
2. Highlight a region of data to be looped for playback. Allow import/export of selected regions.
3. Choose a preset which changes a variety of parameterizations.
4. Link a comment system to a selected region which can save data analysis alongside sonification.
5. Allow user to access a few high level controls, such as tempo and volume.

Prototype Shortcomings

While basic functions were present, the resulting application lacked any robust features for parameterization of data and contained numerous UI bugs. The alpha prototype used general MIDI sounds within Max for timbral changes, which did not allow for the development of specific electronic instruments with modifiable timbres and envelopes. As different data should "sound different" based on its context, general MIDI was passable but did not allow for any low level control of shaping tools such as an ADSR envelope, or EQ filtering. As the aim of design was to avoid the need for any additional software, these features were left out of the prototype.

The visualization element of the prototype was based off the `[function]` object inside Max. (Note: brackets indicate a Max object, which perform a specific function or process in the Max programming environment).The purpose of `[function]` is meant for interaction with ADSR envelopes or other parameter changes. While `[function]` is able to import and visualize thousands of arrays, it is very taxing on the object when its general use is usually from four to twenty points. This led to slow and occasionally inaccurate visualizations due to the scaling behavior of `[function]`.

Figure 2: A typical use of [function] within Max

The region system for looping sections of data uses the object `[dict]`, which allows for JSON data formatting. Each region selection ties to an `[rslider]`, with its size and range tied to the number of indexes inside `[dict]`. Upon making a selection, a "replace" command would rewrite indexes and data arrays within a new "sub-dictionary", which stored that selection for sonification. This proved to be cumbersome for making multiple selections of data and switching between those selections in real time. The main purpose of this system was to preserve the integrity of data by allowing for non-destructive edits inside `[dict]`. *(Appendix A)*

A preset system recalls combinations of timbre, volume, duration, key, mode, tempo, and rhythms, by the press of a button. These presets were not tied to any attribute one might associate with data and were completely generic. The preset system was not tied to region selection, meaning that all separate selections of data all changed to the chosen preset. This was cumbersome for use, as it did not allow the user to tag a selected region with a preset.

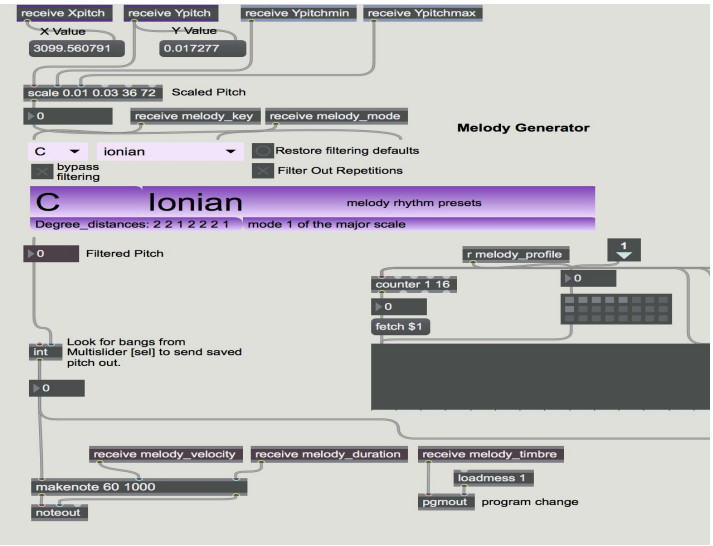


Figure 3: Prototype pitch filtering system

On selection, the minimum and maximum values for both X and Y send to a `[scale]` object. Data is then sonified by constraining pitch values to four octaves, and is then filtered to a key and mode. While data may vary in small floats, filtering created redundancy as many arrays became the same MIDI value upon playback. While a diatonic relationship for sonification was the goal, filtering simply played back "one to one", which created a sense of ear fatigue. Important changes, or the lack of significant change, were not highlighted by the system.

Further inaccuracy to pitch mapping was present through constraining minimum and maximum values to a selection, instead of the entire data set. While a "localized" scaling made for more interesting music, it tended to exaggerate what would otherwise be insignificant value changes when the data is analyzed as a whole.

Timing Issues

Best practice in Max programming involves limiting the number of `[send]` and `[receive]` objects within a patch. The size of the patch and its bpatchers (modular subprograms) necessitated the use of many send and receive objects to control parameterization from a high level. Upon playback, tempo offset would occur due to Max's internal clock. An initial fix for this issue was to use the CPU's soundcard and audio sampling as a timekeeper for playback. The external object `[el.samm~]`, created by Eric Lyon, offered an audio sample rate version of Max's `[metro]` and `[counter]` objects. This limited the amount of latency of different elements within the patch. Since each voice (melody, harmony, drums) ran from its own `[metro]` and `[counter]`, using send and receive objects was problematic as it is opaque what priority Max assigns to data flow behavior. `[el.samm~]` is an older object (last updated in 2013), and became deprecated upon installing Max version 8.01. This created a need to deconstruct each bpatcher and simplify rhythm generation for future use using standard Max objects.

Current Build

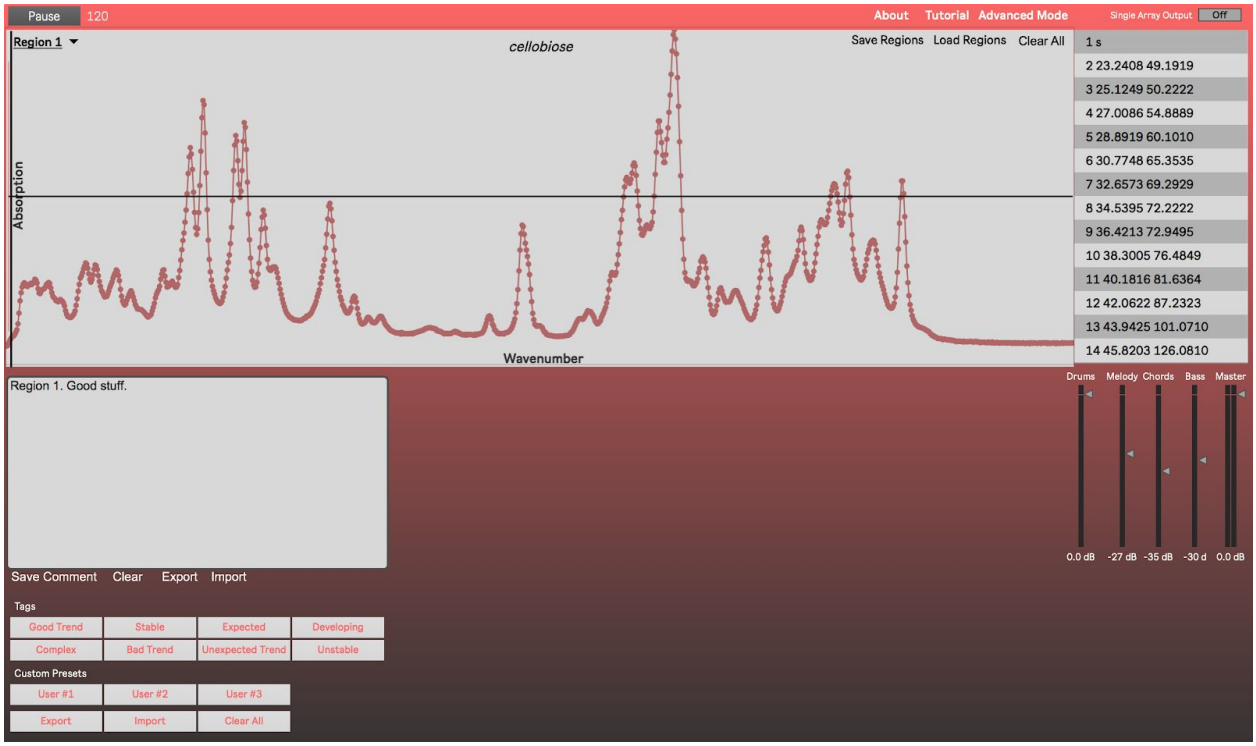


Figure 4: Overview of current build

The Sonify application completed in Spring '19 and has a variety of improvements, most notably specialized processes for sonifying spectroscopy data. This section describes the development of the current build.

Overview

The system uses many sub patchers, or small programs, inside Max to achieve sonification. The process is outlined as follows:

Data Importation (Appendix A)

Spectra files are dragged and dropped onto the GUI for importation. These spectra files are usually a list of arrays such as [181.32 32.95] in a standard text document. Text documents can be dragged and dropped onto the application to import data. To help retain the structure of data upon playback, the text file is first sent to a `[coll]`, which then dumps out all data to be recombined with an index number for each array. This new data is compiled into a `[dict]` object, which allows for more organization through a JSON file format. Upon completion, the `[dict]` is scanned to find the minimum and maximum values for X (wavenumber) and Y (absorption rate) within the entire data set. Min/max for X is used to trigger drum patterns tied to wavenumber, while Y is used to scale pitches to an octave range, key, and mode in playback. The final index is used to scale various UI objects within the patch to match the data, such as a visualization in `[plot~]` and different slider controls used for region selection, which must have the same index range as the `[dict]` object to accuracy create new playback regions.

Region Routing (Appendix B)
When all visualizations and UI objects have been configured with the correct values, selections of data can be made. Selection works by using an `[rslider]` UI object which collects a minimum and maximum value as a list. This list is tied to the same index numbers associated with each array of data. The list is saved as its own separate index inside a new `[coll]` object. When playback is initiated, this `[coll]` object sends the minimum and maximum index numbers as loop points to a `[counter]`.

Playback (Appendix C)
Once the `[counter]` has been fed the appropriate start and end points, playback of data is triggered by outputting data indexes according to a metronome timing. At each metronome pulse, the index is combined with a "get" message and sent back to `[dict]`, which then sends the array out to a combination of melodic pitch filtering modules and chord/bass triggering modules. Each array is pulled from `[dict]` and sonified in real time as it counts through the selection of data. Upon reaching the end of the selection, the `[counter]` will automatically start over at the first index that was selected by the user. Rhythmic variation can be controlled from "masks", or `[multislider]` objects used to effectively mute certain indexes from being output. This feature is bypassed in the optimization for spectral data, where insignificant data changes create silence (thereby making self-generated rhythms out of the data itself). However, a user can toggle between these settings if they prefer to write in their own rhythms for another purpose or context.

Another major change for this module from the alpha was decoupling tempo from presets. Tempo has been noted in other studies (Poirier-Quinot 2016) as an important factor in a user's ability to identify data trends through sonification. Tempo was removed from the "Advanced Mode" window and permanently placed in the main UI.

Filtering (Appendix D)
Pitch filtering occurs in real time as indexes are sent out from `[dict]`. The *EAMIR SDK* and its companion *Modal Object Library* developed by VJ Manzo were used to speed up implementation of this process. Data is first scaled to an octave range, with a default range of four octaves. The user can decrease or extend this range manually for more "resolution", where a larger octave range will provide more pitch changes during playback.

Keys and modes are triggered by the tag preset system or manually by the user in "Advanced Mode". The EAMIR objects save the stepwise relationship of a diatonic key and mode in another `[coll]` object, which is then combined with a "raw pitch" and octave designated from a `[scale]` object. The resulting data is output as a MIDI number. While "pure tones", or non-filtered data as a 1:1 mapping, have more fidelity, the filtering approach was implemented with studies showing that "cognitive organizational overhead associated with atonal systems makes them less well suited as carriers of program information" (Watkins 1985). After all, most Western users are familiar with diatonic tonal systems in general. Additionally, a common issue in sonification is ear fatigue which can arise from unpleasant or overly noisy sonifications of

data. Pitch parameterization and pitch averaging in studies are inconclusive but preliminary results show promise for reducing "noise" through diatonic relationships and pitch scaling in general (Poirier-Quinot 2016).

As a melodic voice is filtered and sent to be synthesized to audio, it is also sent to another EAMIR chord bpatcher which chooses diatonic chords based on each note received. The chord filtering system may produce a 1:1 relationship between the melodic voice and accompaniment, or can "look ahead" by reading arrays from `[dict]` in non sequentially. This feature was implemented to increase the feeling of arrival between the relationship of melody and harmony. Our common expectation of tension and release in Western music can be heightened by the offsetting of a melodic voice from its accompaniment chords.The EAMIR chord object also has affordances for fuzzy logic, where a single note can trigger a variety of related diatonic chords that have some relationship to the base pitch. This feature is not currently implemented in playback but can be easily set up from the main Max patch if a user is not working with the standalone application.

In data specialization mode, chords are triggered by the pass of an apex of data to signify molecular activity from a spectrometer reading. The threshold by which the system identifies a peak can be manually changed by the user, but defaults to an increase and decrease by five scale degrees in pitch. As a peak is passed, a `[gate]` object opens which triggers chords based on the metronomic timing in the playback module. As a default, chords are triggered in 8th notes. An interesting phenomena that occured from this implementation is that more significant apexes from peaks tend to trigger more chords in general, allowing the user to hear how big the peak is by listening to chordal activity. A more detailed description of this design is elaborated on in the interview notes.

Lastly, the root note is pulled again from chord output and is dropped an octave as a bass voice. The bass voice has its own unique rhythm per preset, and can be heard as an accompaniment even if peaks have not been reached to trigger chords.

Synthesis and Digital Signal Processing (Appendix F)

Once all data has been assigned to MIDI values corresponding to an octave, key, and mode, it is sent to a bpatcher that contains a polyphonic version of a BEAP oscillator. The BEAP package within Max was used to shorten development time, as it contains fully functioning synthesizer modules commonly used for sound design. A polyphonic version of the "bp.Oscillator" bpatcher was created that affords common synthesizer parameter changes such as waveform, ADSR envelope, filtering (lowpass, hipass, bandpass), and octave changes. To extend the timbral possibilities, two oscillators were combined in the `[poly~]` patch with the same functionality, allowing for combinations of waveforms. The bpatcher contains 30 custom presets which may be used for any voice in playback, and are attached to each filtering module to create an audio signal. More presets can be created by users using the bpatcher located in the Max source code.

Each bpatcher is sent to a `[live.gain]` object for localized volume control, and then sent to a "pre-master" gain object that combines output into a single stereo signal. The premaster gain

object connects to a reverb bpatcher based off the "yafr2" effect. Reverb can be bypassed altogether by toggling a `[gate~]` object, or can be combined as a wet/dry signal. Lastly, the pre-master signal is then sent to a"master" gain which controls volume of all signals simultaneously during playback.

Drum sounds were created by mixing down samples from a classic Roland 606 drum machine instrument in Ableton Live. These sounds are imported through the `[playlist~]` object, and are triggered through rhythm masks tied to the playback module.

Presets and Tagging (Appendix G)

The Max `[preset]` object stores all global changes to the system, which are swapped through a tag system. Triggering a preset will change the key, mode, accompaniment rhythm, and timbres of synthesizers. Each preset is tied to a tag associated with a data attribute, such as "expected trend" or "developing". Tag names can also be swapped to generic names from the Advanced menu, which may be beneficial for educational settings or exploratory learning. The default data presets were designed to represent data phenomena by tying common western tonal harmony practices to parameters. For instance, the "Good Trend" tag triggers playback in C major, with a steady rhythm, and more "pure" synthesizer presets using sine waves and triangle waves. A "Bad Trend" tag will execute noisier waveforms created with square and sawtooth waves, a key and modality of A minor, and more syncopated rhythms.

The point of mappings were as follows:

1. Allow an expert to describe data phenomena, and without musical knowledge, tie that data phenomena to some musical phenomena.
2. Create distinct musical differences to differentiate selected regions for playback.
3. Increase accessibility by hiding more complex sound design and compositional processes to increase immediacy in interacting with data.

In "Advanced Mode", users can also create up to three custom presets by accessing the same global controls used to make presets in the tag system. These can be exported and shared with other users.

Tag Features for Future Development (Appendix H)

There is an abstraction within the tag system for logging multiple tags at once, and swapping between presets according to a beat/bar timing system. In future development, tags may trigger only a few parameters, allowing for the combination of attributes to represent data phenomena. This feature was removed due to bugs in the `[poly~]` synthesizer patch, which may require a more robust design to take full advantage of the process.

Comment System (Appendix I)

Used for logging observations of data, the comment system is tied to a region selection. Comments are saved through a `[textedit]` object connected to another `[coll]`. Comments may be imported/exported as .txt files.

Wavelength Mapping (Appendix J)
Scales X value (wavelength or wavenumber) from spectra data to trigger a series of drum presets. These drum presets help to aurally indicate where absorption rates are occuring by three distinct regions.

Advanced Mode
Shows all features that are hidden by default to be used in lower level modifications to data and musical parameters. All UI objects to control aspects of sonification (pitch, timbre, rhythm) as well as data behavior such as peak detection, are nested in advanced mode. The purpose of this was to not overwhelm the user with too many options, but provide the support and capability to customize aspects of the system. These features are important to future studies, where the system may be optimized for other data types, or used in a combination with other systems to study best practices in sonification design.

2. Interview Process

A series of interviews took place to obtain Professor Timko's input on how to optimize the Sonify system for spectrometer data by implementing domain-specific knowledge into the sonification model. We held interviews in the second year of development between January and April 2019. Before each interview, Professor Timko was able to use a current build of the application for a brief period of 10-20 minutes. Each interview focused on the parameterization of spectra phenomena, and UI design for ease of use in educational settings per his criteria. Due to the nature of this iterative design process, results are partially embedded in each interview section. A summary of all results resides in the results section. I asked Professor Timko the guiding questions below, and took summative notes on his responses." Full note cards are available in the Appendix.

Interview #1 (Appendix K)

We held an hour long interview where Professor Timko used the application for twenty minutes before discussion, after which a series of questions regarding specializing the system towards spectra data took place:

1. What activity determines the compounds of a molecule?
2. What is the phenomena of absorption rate during a spectrometer reading?
3. How can that absorption rate be better represented through music?
4. If X represents wavenumber in spectra, what parameterization could give the listener a better understanding of where they are in the reading?

Timko's responses to this questions are summarized as follows:

Both X (wavenumber) and Y (absorption rate) relate to molecular vibration. The bonds of the molecule move in different directions and patterns based on when and where heat hits the

molecule. The wavenumber, or where that vibration occurs in the molecule, is what determines the composition of the molecule. For instance, activity from wavenumbers 0-1000 indicate a fundamental, or "skeletal" bond. These do not move as freely as other bonds can. The peaks in the readings should emphasize this, as chemists do not often pay attention to the individual values in a reading. The peaks are the most important information for quickly identifying molecular structure.

At this point in development, the application was converting every array into an audible pitch, regardless of the significance of a change absorption rate. By scaling this data to diatonic pitch, the listener would hear many repetitions of the same note if a large enough change had not occured in absorption rate. This design adds unnecessary “noise” to the sonification. For peak detection, the listener could potentially identify peaks by listening for increases and decreases in pitches. This required far more active listening over time to get a sense of the molecular activity. Timko suggested to deemphasize individual arrays, and to try to emphasize peaks to the listener.

Upon discussing the vibration of molecular structures, the case could be made that mapping pitch to absorption rate is an appropriate mapping. An example in Oregon State University’s organic chemistry resources visually maps spectra to vibrations within the molecule, describing it as “Every line in an IR spectrum arises from activating a molecular vibration. In the "functional group region" these are generally simple: only one bond, or a collection of two in concert. In the "fingerprint region" these tend to be more complex combinations of bond vibrations.”

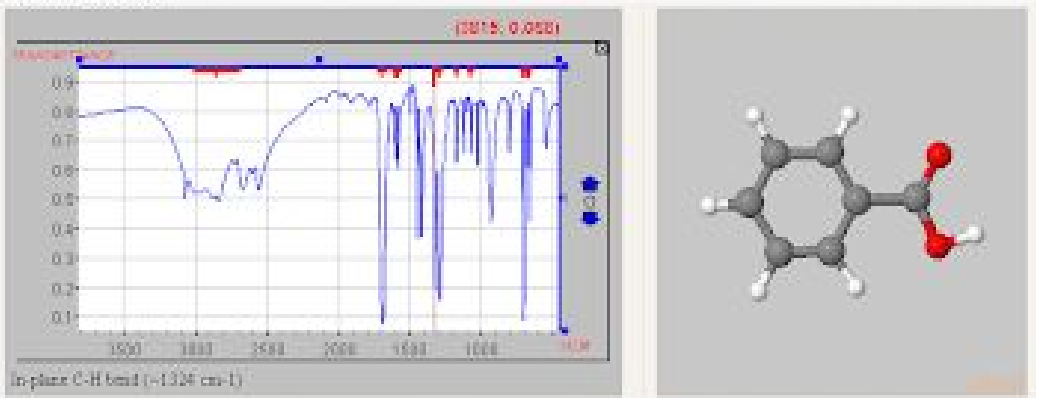


Figure 5: An example of vibrations by wavelength of Benzoic Acid via https://www.science.oregonstate.edu/~gablek/CH335/Chapter10/IR_vibrations.htm

As a heat source hits the molecule, different sections of the molecule vibrate at various speeds. The “functional group” denotes a structural, single bond or possibly two. This earlier region in wavelength is the “skeleton” of the molecule, while the “fingerprint region” is more complex combinations of bonds that make up the molecule. The analysis of X (wavelength) by Y (absorption rate) indicates the potential makeup of the molecule.

From this information, I modified to the system to emphasize peaks. Using a simple operation in Max, the system detects increases and decreases in values over time by a threshold value. To test

the system, a default value of 5 (up and down) indicates the passing of a peak in the data. The threshold can be modified by the user to control the sonification behavior. As the apex of a peak is passed, diatonic chords are triggered indicating activity. These chords are tied to the individual array being sonified as a melodic voice to create an accompaniment.

Silence in music is a form of distance, and duration and magnitude of silence has an effect on our experience of compositions. By tying peak detection to chord triggering, each data set also becomes unique in its form -- no region of data will sound alike in chordal accompaniment. The general sense of chordal activity can also indicate peak activity. The sonic events tied to peak activity were considered acceptable upon a third test and interview.

Interview #2 (Appendix L)

The second interview focused on indicating wavelength/wavenumber values to the listener through an aural signal, and simplifying the interface for use with students, who receive support in a class or group setting with Professor Timko or another experienced chemist. The series of questions for this interview were as follows:

1. Now that peak activity is aurally indicated, how can we also indicate to the listener the location of this peak activity via wavelength/wavenumber values?
2. What affordances are useful to users, and which features can be hidden?
3. Are there bugs or behavior in the program that are problematic for use in an educational setting or study?
4. Are the tag system presets varied for attributing musical properties to a selected region of data?
5. Are the presets pleasing to listen to?
6. Are tag descriptors useful for describing data phenomena?

Our discussion is summarized as follows:

While the new sonification behavior helps to signify peak activity, it does not solve the issue of indicating the location of a peak in terms of wavelength. Since we know that wavelength corresponds to what type of bond is likely vibrating, and this is important to identifying molecular composition, another sound source must be layered with chordal activity to indicate location. Previous methods in an undergraduate prototype of a sonification system developed for Professor Timko tied wavelength to MIDI velocity, or volume, which was problematic.

Volume for location is considered natural in a physical space using acoustics. We can use the inverse square law to calculate that a sound loses 6dB amplitude as a distance is doubled (Cooper, 2019). In a virtual system with no spatialization features, this mapping is too abstract. Humans can detect a change of 1dB, or 12% of amplitude increase from a soundwave (Smith 1997). However, asking a listener to quantify amplitude from listening to music is difficult. While we can generally assume a normal speaking voice to be around 50dB, and an airplane passing overhead as 150dB, the discreet differences of volume are not apparent to even practiced

musicians and sound engineers without signal detection from microphones and Fourier transform processes.

Based on this discussion, I further explored other options for sonifying X (wavelength). We considered two methods as alternatives to a volume parameterization. The first method was to tie wavelength to the structural elements of a longer form composition as sections. Wavelength values can be divided up into three key regions, from 0-1000, 1000-2000, and 2000-5000. These wavelength regions could tie to traditional sections of composed works, as an "A section" or "B section". Parameterization would involve tying accompaniment chord progressions to X values, so that 0-1000 becomes "A section" by having a particular progression that always occurs. The "A section" could emphasize a standard tonic of a diatonic key, while the "B section" could emphasize a subdominant, dominant, submediant, or other chord quality.

Our other approach was to leave chord accompaniment tied to Y, or absorption rate , and to use drum patterns to signify a region of wavelength values. Drum patterns may be easier for a non-musician to differentiate than diatonic chord progressions, which usually require some formal training to identify. Altering the global tempo of playback could create confusion to the listener, while subdividing drum rhythms over time may not. As an example, 0-1000 wavelength becomes drum patterns that emphasize quarter notes, and then 1000-2000 emphasizes eighth notes. The listener would then have an aural indication of what wavelength the peak activity is occurring in without seeing any visualized data, and without understanding music theory.

I implemented the drum pattern approach using the major wavenumber regions identified by Professor Timko (Appendix J). Drum pattern presets emphasize smaller subdivisions of rhythms by region. 0-500 produces no drum patterns, 500-1000 emphasizes quarter notes, 1000-1500 introduces eighth notes, 1500-2000 further emphasizes eighth notes and introduces sixteenth notes, and so forth.

We also discussed user affordances, and how the UI could better facilitate informal learning settings. Timko proposed reintroducing parameters to the user such as a volume visualization from `[live.gain]` objects. We also wanted to de-emphasize the comment section which was not the primary concern for engaging with the data sonification. Other UI changes such as a feature that pauses playback when dragging and dropping a new file into the program prevented crashes during testing.

While the tag presets sounded acceptable to Timko, he noted that the data descriptors may be confusing to users in a STEM informal learning environment. We decided to include a toggle in "Advanced Mode" which renames all tag descriptors to general musical preset names. This allows Timko, or another user, to modify how users see the UI and tag descriptors as necessary.

Interview #3 (Appendix M)

The third interview followed up on desired features and bug fixes discussed previously, most notably a wavelength parameter mapping. Questions then moved on to discussing potential

future features to implement within the development window. The questions were proposed as follows:

1. Is the parameter mapping of wavelength to drum rhythm changes interesting, and does it help to indicate a general location within the data set?
2. If not, what changes could better portray wavelength values?
3. What other features, if any, would be useful to have?
4. How may the system be used beyond the scope of this initial research and development?

Discussion from Interview #3 is summarized as follows:

Timko found the drum percussion parameterization to be acceptable, but had concerns specifically with the presets that were triggered. In my initial approach, the 0-500 region of wavenumbers produced no percussion whatsoever. While this intended to make it obvious where the user was hearing absorption rates, Timko found it would be boring for an extended period of time to listen to, especially with younger students. We discussed how popular music has a "build" and how foundational rhythms tend to elaborate over time. A simple change proposed was to use a "four to the floor" kick drum sound as this foundational preset. Timko reiterated that this 0-1000 wavelength region is foundational to the composition of molecular structure. Presets were modified to reflect this need.

We further discussed how many drum presets should trigger by a wavelength region, noting that too few presets was uninteresting, and too many could be confusing to the listener. Timko also noted that the subdivision of the three main wavelength regions were arbitrary, and that more subdivisions could provide a more informative and interesting listening. The three regions previous proposed were then divided into eight regions covering every 250 wavenumbers up to 2000.

A following discussion on future improvements or "would be nice" features followed. The most desired feature for Prof. Timko was the ability to export the sonification as a music file to share, or to create longer form pieces of music using multiple data readings. What we decided was most significant for future work was the ability to showcase sonifications as more than an analysis of a single set of readings.

For example, we explored the options of longer pieces of music to illustrate the differences between organic sugar compounds and artificial sweeteners. To promote education on the risks of different sweeteners to public health, a musical piece could allow the audience to see and hear the discrete differences between the spectra of these compounds. Since a product like splenda (sucralose) is visually indistinguishable from another sugar, like fructose or glucose, a user could use a multimodal representation to become aware of these differences. The ability to export regions of each data sonification and string them together became apparent as a future consideration for development. Particularly, sonifications using the same presets as a control would allow the user to better hear the differences in molecular activity across multiple data sets like splenda and fructose.

A limitation in Max's object library to "offline" export music files identified potential issues in creating music files -- the `[sfrecord~]` object could record playback, but that recording must occur in real time. The sonification could be bulk exported as a MIDI file, but it would not retain any of the signal processing from the application. The application must be able to function as a standalone with no additional software. Exporting only sections of data was the most typical use of the system and supports the current architecture of the build. For future development, Prof. Timko noted that being able to import multiple data sets for sonification simultaneously would be a useful feature, but was outside the scope of development at the time. Notes on approaching this feature are elaborated on in the discussion section.

Another difficulty in using the Sonify app is the ease of access to spectrometer readings. While many scientific websites presented by Prof. Timko had many images of readings, it was not easy to get text documents of arrays. Some websites required subscription fees to access this data. We discussed adding a "baked in" set of spectra data for users to experiment with if they did not have access to data either on the web, or directly taken from a spectrometer. Data collections from graduate students at WPI who already create spectra for their research may enhance the database in the future.

3. User Study

A pilot user study evaluated the application's accessibility. This evaluation was important to ascertain its viability in future tests in educational settings as per Prof. Timko's use scenarios. 27 undergraduate students at WPI participated in the survey. Because this pilot focused on accessibility, the survey did not ask for a user's level of familiarity in music technology or chemical engineering. Participation in the activity was optional to students, and all submissions were anonymous.

Users tested the Sonify application in a music technology classroom on campus using iMac computers and Sennheiser closed-ear headphones.. A Google form survey contained video tutorials of each section of an embedded tutorial within the app. Upon finishing each video, users execute the same procedure and then report on a scale the level of difficulty in doing so. Each question is standardized as "Please rate your experience with.." followed by a linear scale input from 1 to 5, with easy being 1 and 5 being difficult. Detailed analysis of the findings are included in the results section.

Results

The section will outline results based on each section of the methodology in three parts, concluding with a final summary of all results.

1. Summary
2. Development & Interview Results
3. User Study Results

Summary

The intended goal of the methodology was to provide a system that sonifies spectroscopy data in the Max programming environment, which could then be validated through testing with an expert in chemical engineering at WPI, Associate Professor of Chemical Engineering Michael Timko. The development process was extensively documented in the methodology and appendix, while the source code was made available on a public facing WPI website at http://sonify.wpi.edu. This website contains all documentation, source code through a github repository, and downloads for a standalone application on both Windows and Mac platforms. Features requested through interviews were implemented in the iterative design process, which optimized the system for spectroscopy sonification.

Following the development process, a pilot study provided initial feedback for future studies and development by looking to survey the accessibility of the application. These data suggest the application is ready for more comprehensive studies in the future, but could benefit from improving UI design in some areas.

Development & Interview Results

The interview process led to many new specializations for the sonification of spectra. Most notably, the system produces much less “noise”, as in unnecessary or distracting aural signifiers, using a peak emphasis for absorption rate combined with the removal of pitch repetitions. As noted earlier, peaks in spectra tend to indicate the vibration of bonds within molecules, and that activity determines the makeup of the molecule. The triggering of chords upon reaching the apex of a peak gives a unique voice and process to indicate this data phenomena. This feature additionally creates unique sonifications through repetition removal, creating a sense of different rhythms and periods of rests as the algorithm deemphasizes insignificant data changes.

Wavelength, or the X array of data fed into the system, maps to percussion rhythms as a result of the interview process. Where X was previously tied to volume/amplitude in an undergraduate version of a sonification platform at WPI, percussion provides a more aesthetically pleasing

listening experience while still notifying the user of the location of absorption rate within a data trend.

Resulting UI changes from interviews revolved around accessibility for younger STEM students (around middle school age) as per Prof. Timko's use case scenarios. The main intention was to make the application "rugged", so that it could not be easy broken, as well as accessible. Further use cases with a younger audience is noted as a future study in the discussion. An outline of accessibility changes are detailed as follows:

- **Tutorial Window**
  Provides more detailed explanations on a pop-up window of how to use the application.
- **Hint System**
  Explains each UI element on mouse-over.
- **Advanced Mode**
  Hides all customization features including rhythmic and timbral changes, signal processing effects, and spectroscopy specialization processes. Also allows the user to change the name of tag attributes.
- **Custom Presets**
  In Advanced, users can create their own presets and save them as a .json file. These presets trigger timbre changes on synthesizers, key and modality, or even volume settings.
- **Playback**
  When pressing space bar, playback always starts at the beginning of a selected region. Users can bypass this by holding shift before pressing spacebar to continue playback at the playhead.
- **Export as .wav**
  Toggles real-time recording of audio signal generated from sonification.
- **Bug fixes**
  - *Crashes occurred while importing new data during playback.*
  - Tags sometimes did not apply a preset change until pressed more than once.
  - Removed "multitag" process to prevent volume spikes when changing presets.
  - Fixed a bug where the visualization randomly moved the location marker when reloading the application.
  - Resolved an issue where entered comments were not tied to region selection.
  - Decreased the size of the comment field and included volume attenuation visualization in the basic display.
  - Added a pulsing "loading" icon to inform users when data was finished being compiled.
  - Allowed window resizing in standalone application to fix display resolution differences between systems.

User Study Results

While 27 undergraduate students as a test group confines the test to a particular age group, it proved sufficient for a pilot study. It is reasonable to expect that certain kinds of computer workflow and interactions are familiar to most students, and thus, part of the ease of accessibility may be biased in this age group. A follow-up study with a greater diversity of users may yield different results. In summary, this initial pilot shows the program is accessible and ready for use in more comprehensive studies, as well as educational settings as per Prof. Timko's intended use. Out of the 27 students who used the program, 50% or more found all of the procedures rated "1: Easy", except for advanced features, which 44% of students described as easy. The most reassuring result was the design of the tagging system, which 74.1% of students found to be easy to use. This design choice meant to simplify the workflow of sonifying data by removing or hiding lower level musical parameterizations through presets. Advanced features scoring the lowest in ease of use ties back to the tagging feature, in that hiding lower level functions may create a more accessible and enjoyable experience with the application.

A few findings highlight potential UI improvements for the future, which are elaborated on in the discussion and conclusion sections. An example of collected survey data is shown below. The full collection of survey results are listed in appendix L.

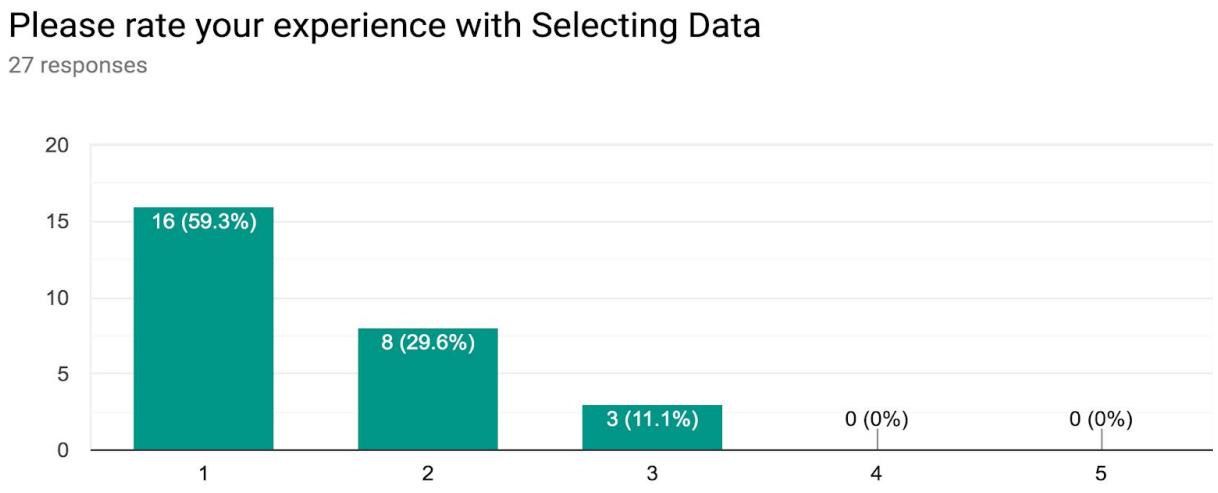


Figure 6: Selecting data results. Reported on a scale from 1 "easy" to 5 "difficult".

Discussion and Outcomes

Development Discussion

The development process shows that sonification systems can be designed and modified to accommodate a wide variety of data in a specialized manner through a modular system. While expertise in lower level programming languages such as JavaScript may be beneficial for implementing more robust systems, particularly web-based systems, the Max programming environment is conducive to collaboration between musicians/creatives and the sciences. Musicians are familiar with the visual object oriented design of Max that affords quick methods to manipulate data into MIDI or audio signals. For professional use, it was expected and confirmed that a chemical engineer could quickly use the program to sonify data. Each use in every interview session for Prof. Timko's interview enabled him to import data and create a sonification in minutes.

While all sonification mappings (and visualizations for that matter) are abstractions, close collaborations with experts in scientific fields of study can promote the building of multimodal representations of data that are context specific in nature. These collaborations enable domain knowledge transferral across disciplines, which can help to contextualize the nature of a sonification system. Mappings were misguided following the first interview in the design process, despite being grounded in music theory and sonification background research. After reflection and dialogue with a chemical engineer, a better system emerged to meet both musical and scientific needs. Collaboration is essential to improving sonifications in the future if they are to reach a higher adoption rate in a variety of settings such as educational or professional.

It should be noted that the primary use case following this development would be in a pedagogical context, such as a STEM module. The application could also be used to promote and raise awareness about spectroscopy in education as a whole. The pilot study only accesses the ease of use to which students are able to perform the processes required to create a sonification in the application environment. While these studies were positive, it is important going forward to test and further modify the system to obtain more cognition data on how sonification may enhance chemistry education through multimodal representations of data. Multiple limitations in this research, from project scope to time, made this kind of work unfeasible. In future research, I recommend combining consultation from both the chemical engineering and school psychology fields, which would greatly benefit coming to new understandings between sonifications and pedagogy. This study and consultation would require more iterative development based on results, so it may still require collaboration with an expert in sonification.

From a musical perspective, the current Sonify model comes close to western tonal tradition, and further research and design would help answer if incorporating cadential structures would bias the listening experience to a degree that the significance of data is lost. The program lacks cadences we associate with harmonic structures – such as a half cadence ending on a dominant chord that proceeds to another progression eventually returning to the tonic. The current model either triggers a diatonic chord based on the mapping of a single array from the spectra, or offsets this process by looking ahead in the data. While the interview process comes to conclusions that

highlighting and emphasizing the peaks of absorption rates through chord triggering is sufficient for spectra sonification overall, the user is simply listening for chordal activity. For an even more aesthetically satisfying experience, the size and frequency of peaks could be tied to cadential structures, so that a series of peaks trigger a plagal cadence, and another trigger a deceptive cadence, and so on.

It is important to note that such a cadential structure may bias the listener's interpretation of peaks. A study comparing the differences between a harmonically neutral system and one with clear cadential structures would be a worthy investigation, or more studies confirming or denying that Western tonal systems (used with a Western audience) are more beneficial than atonal or microtonal systems. There are studies on the comparison of sonification programs in the context of aiding visually impaired users. Their findings suggest that visually impaired users experiencing sonification are more likely to justify and understand parameterizations, where sighted listeners found mappings to be more abstract. A visually impaired listener upon hearing a mapping of frequency:dollars in the study "...explained her responses by noting that a coin dropped on a table makes a high-pitched clink, whereas a roll of quarters makes a clunk, and a bag of coins makes a lower-pitched thud, leading to the negative polarity for the frequency: dollars mapping" (Walker & Mauney 2010). Interestingly, this mapping was the inverse in frequency that the user justified. Such a finding further implies that the experience of parameterizations vary across diverse groups, where the most obvious mapping may even be the opposite of what users may wish to experience.

Tying musical schemata to molecular phenomena directly may prove difficult without a more robust system to analyze data upon importation. The design focus of Sonify was to allow an expert to "mark up" the data with tags, which trigger musical events. This bypasses the need for the program to make its own decision on what parts of data to emphasize. Another interesting future study would be allowing the system to suggest what parts of the data to highlight before a user makes a manual selection of a region of data. This could be done through a similar process as what currently triggers chords from peak activity in absorption rates, but would have to be implemented in the beginning of the data processing chain upon importation. This importation process is already taxing on the system in Max—it currently takes anywhere from three to ten seconds to successfully analyze and map data—and this change would likely further slow down the process.

Additionally, while outside the scope of this initial research, it is important to consider comparing the Sonify application to other sonification systems, whether they are multi-purpose or specialized to process spectra. By looking at the affordances and constraints of a variety of systems, the field can garner a better understanding of the parameterization process and user interaction from a HCI perspective. The initial pilot study gives some insight into how accessible the Sonify system is, but further research is necessary for how that extends into more diverse audiences, or other use case scenarios such as installations or an education curriculum.

For instance, the aforementioned the Data-To-Music API developed by Tsuchiya, Freeman, and Lerner also considers harnessing the power of hierarchical musical frameworks through higher

level control. The study similarly raises the question of how "organized sound" may impact the transparency of data (Tsuchiya et.al 2015). Both Sonify and the Data-To-Music API take the approach of creating "music structure models", or models which control a variety of parameters within a musical framework simultaneously. The Data-To-Music API models create abstractions of these structures, which control units such as "...rhythmization, note dynamics, articulation, pitch scale, chord voicing, timbre modulation, and so on." Sonify has specialized processes for spectra data, but follows a similar design aesthetic that the user/expert can create their own parameterizations from a higher level process. In this way, the two programs are very similar in this regard, however Sonify leverages specializations for spectroscopy data as well.

A tradeoff not mentioned in the method of design earlier was the decision for the application to run independently by not requiring any additional music software to run. One of the drawbacks of this design was having to develop synthesizers, sequencers, and other musical processes directly in Max. These modules were sufficient to achieve an independent sonification platform, but they are somewhat limited in the scope of sound design. It should not be assumed that any user has access to specialized VSTs or DAW applications when they use the platform, and that a goal of mass adoption is predicated on independent applications. However, VSTs offer many more affordances for sound design, such as complex wavetable synthesis, more realistic reverberation or spatialization, and sampling functionality, and do so more efficiently than systems in Max without designing on a lower level using `[Gen~]` or working with Fourier transform processes. A tradeoff may be to build future versions of the system to work directly inside another DAW such as Ableton Live via Max for Live, or to offer plug-in support directly within Max combined with MIDI and audio out. The application should offer both options to remain independent of third party software, while allowing for more experimentation in sound design.

Pilot Study Discussion

While results confirm the application is ready for more comprehensive studies in the sonification field, some processes in the application may need adjustment to enhance accessibility. Possible modifications based on findings are elaborated on below:

1. **Selecting Data**
   59.3% of users found data selection to be easy, while 29.6% rated the activity a 2 as fairly easy. 11% of users rated it a 3, as somewhat difficult. The original design for selection did an automatic iteration of region selections. As the user clicks and drags to select a new region, the application automatically would advance to the next region. Currently, users must manually choose a second or third region from a drop-down menu. This design choice was made to ensure that selections correctly entered a coll object in Max. However, the previous model of automatic iterations may reduce the amount of steps necessary to make selections for sonification, thereby simplifying the selection process.

2. **Saving and Loading**
   51.9% of users found this procedure to be easy. However, 33.9% rated a 2, 11.1% rated 3, and 3.7% rated it a 4 as fairly difficult. Due to development time constraints, a consolidated file saving system was not created. While these results are largely positive, the current design asks users to save different sections of the application as separate text documents. Future development should include a system that saves all user settings (data regions, presets, comments, tempo, etc) as a single file. This file should have a custom extension (such as data.sonify) to avoid any confusion, and should save directly to the application's folder as a default.

3. **Advanced**

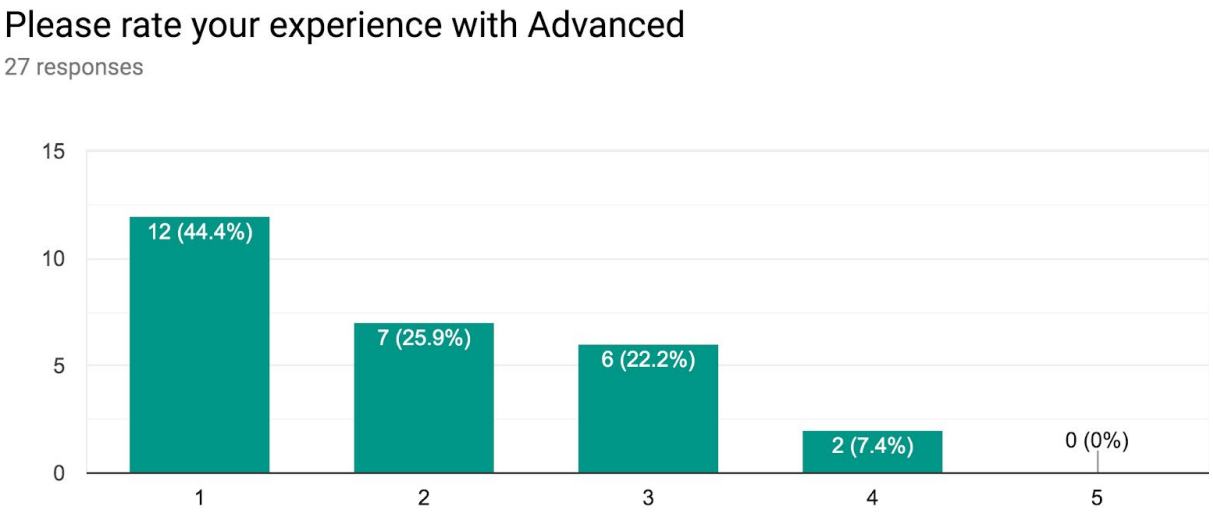


Figure 7: Advanced results

Initial findings show this is the weakest part of the application's design, with more than a quarter of users describing its use experience as somewhat difficult. One user during the study noted that UI objects such as the multislider only respond when being clicked on the upper half of the object's window. Certain parameters, such as selecting new presets, are displayed using only integers instead of drop down or scroll menus with detailed descriptions. Even lower level controls such as using the BEAP polyphonic synthesizer module are hidden to the user, but may be more beneficial to include for customization in the future.

Another issue to note with this finding is that the tutorial for this process merely explains what controls are available to the user, and briefly demonstrates clicking them through the video tutorial. The tutorial does not explain why or how you would want to use these lower level controls. An in depth "advanced tutorial" may need to be included to better introduce affordances to the user. In defense of the poorer score for this procedure, it strengthens the argument to reduce interactivity to higher level control through the tagging system, which scored very high on its ease of use.

Conclusions

This research has resulted in the development of a system specialized in spectroscopy sonification through collaboration with a chemical engineering expert. What is novel about this system is its combination of the three common design paradigms, discrete-point,continuous, and model-based sonification approaches. Users hear continuous data through the melodic voice, they hear specific triggering of audio events with a discrete-point design, and users may choose to impose different models upon the sonification as a whole through the tagging system. The combination of these approaches is flexible and worthy of further investigation given that many sonfication models are compartmentalized to one or two of these approaches.

The aforementioned absolute mapping of time to wavelength and pitch to absorption rate is sufficient, but does not include any models for a musical or "organized" sonification in the way Sonify or the Data-To-Music API does. What makes Sonify differ in in its approach is the combination of specialized parameterization processes and higher level customization by the end user. Sonify combines these "hard coded" parameter mappings (such as triggering chords on the apex of peaks of absorption rates) that make a broad analysis possible with the "musical structure models" of the Data-To-Music API approach, allowing a user to tag data and apply a variety of processes to further describe the data phenomena in a musical way. While agnostic models that use one-to-one mappings are useful, and perhaps preferable depending on the circumstances, Sonify offers musical affordances at the user's own discretion through the tagging system.

In a sense, the Sonify user dictates the level of specification in regards to the behaviors of parameterization through higher level functions. As the visually impaired user noted for a mapping of frequency to dollars, it made more sense to them to invert the pitch relationship so that a larger sum of money was a lower pitch instead of a higher one. These relationships can be altered with the press of a few buttons on the Sonify platform without impacting any of the more specialized features used to denote absorption rates as a whole, or how to identify a wavelength region. The only entirely "agnostic" design in Sonify is its timbre controls, which do not make any attempt to symbolize tradition instruments. This feature, as it were, does not imply to the user that what they are hearing is entirely traditional music, as they may be inclined to think if the resulting sonification was comprised of orchestral strings and pianos.

This design meets its intended focus of being used as an educational tool for Prof. Mike Timko

through its parameterization processes, which were aligned to spectra phenomena. Its current state enables further research in developing its use in a broader distribution, or modifications to the source code to parameterize other forms of data. For example, the Sonify system could be used to create real-time scores in a notation format from the same data input for a musical performance following some modifications to the source code. Additionally, the pilot study reinforces that the application is currently accessible and easy to use, however, further research with a more diverse user base is needed to test its viability with younger audiences if it is to be used in a STEM education context.

While time constraints hindered the development of a fully modular system, the current work is ready to be modularized and distributed as a Max "sonification toolkit" package. This package would aid in future sonification development. A full breakdown of potential future directions for the project are elaborated on in the following section.

Future Directions

Most importantly, while all specializations to sonify spectra were created in collaboration with a chemical engineering expert, cognitive studies could confirm or deny if these specializations impact the recognition of data phenomena in regards to pitch filtering, peak emphasis, and wavenumber location through percussive rhythms. It is likely that these specializations will need modifications as more comprehensive user studies that focus on how parameterization affects understanding are tested. However, for the current purposes of the application, these features are sufficient for their intended use in Professor Timko's work.

There were also suggested features during the interview process that could not be completed due to time constraints. For self-directed or informal learning environments, it was noted in the interviews that the focus of study may phase between spectroscopy concepts and data exploration to music and composition, especially with middle school aged students. The current platform behaves more like a tool, but it can easily be modified to emphasize musical concepts if necessary. Many specialized features that were implemented for spectra sonification already have toggles that can deactivate certain parameterizations. For future work, more parameterizations should have this toggle as more features are introduced. This would allow a user to tailor the sonification system to their needs.

The most compelling feature to add in the future was agreed to be the ability to import multiple data sets into the program simultaneously. The use case noted in the aforementioned third interview describes a composition that sonifies many kinds of sugars to teach about the impact of sugar consumption and the dangers of synthetic sugars. Currently this concept would have to be manually pieced together in a separate digital audio workstation. Expanding the current sonification model to handle multiple data sets would require duplicating many of the modules already available within the system. However, a better GUI for the user would be necessary to interact with that much data, as the current base Max objects used cannot support visualizing and selecting multiple data sets.

Many processes would benefit from custom UI designs and websocket implementation. For example, a faster and more accurate data visualization to accompany sonification is needed for future development. The [JSUI] object, combined with Node.js, may help facilitate this need. Due to the time restrictions in this project, a custom UI was not created. Conversely, using standard Max objects guarantees compatibility across all development platforms and versions of Max, which avoids bugs in building stand alone applications as well as installing multiple custom packages to program.

Additionally, the ability to pull spectra data directly from a web server would make exploration of data more accessible to users by removing the need to host files locally. Spectroscopy data may not be accessible to end users, so either an API function or a server which hosts data would enhance the experience for novice users.“Node for Max” introduced in Max version 8 would likely assist in creating these features by assisting in porting Node.js code directly in Max.

Max also affords many ways to modularize programs into bpatchers, which are self-contained patches that can be quickly added to projects to facilitate development. Due to timing issues with Max’s internal clock, many bpatchers were deconsolidated back into the main patch for Sonify. However, most of these issues can be avoided in future builds by avoiding send and receive objects. Ideally, the main components of the patch should be offered as separate bpatchers for other developers in the future. For example, the entire data processing abstraction could offer a quick way to always import data arrays and scale them to MIDI values. These bpatchers could be offered as a package via Max’s package manager feature. Users could then combine sections of the Sonify platform to fit their own sonification needs in Max, thereby potentially decreasing development time on other projects.

For research, collaboration with the manufacturers of spectrometers could result in a direct connection between the hardware and Max, allowing for immediate data importation upon producing spectra. The immediacy of this outcome would greatly aid sonification research in the future, and provide a compelling installation piece. This kind of work would likely also require either an existing chemistry program who has access to spectrometers, or a grant involving a specialized development by a spectrometer manufacturer, to implement such a design.

Appendix

Includes more detailed explanations of each section of the patch and the processes to achieve sonification. These notes may be useful to other developers who may wish to modify the source code of the project or implement similar solutions in their own work.

Appendix A: [Dict] and Data Processing

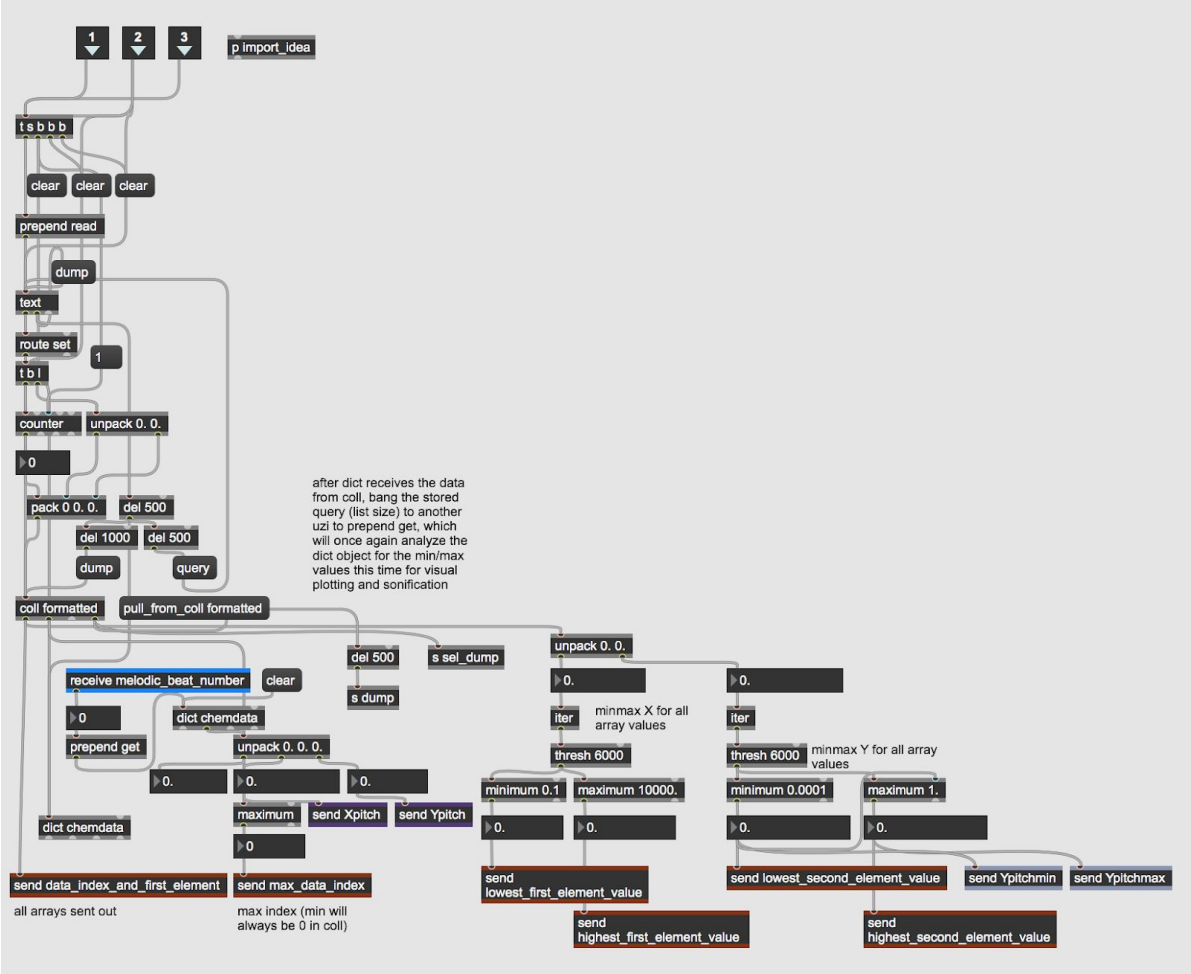


Figure 8: Data processing sub-patch

This abstraction takes any text file of data arrays and processes it for sonification. Files are imported through a [dropfile] object, which triggers a full clear of all objects that store data in the patch before loading in the new file. The data is then sent to the [text] object, where it is dumped back out alongside a [counter] object. The [counter] is used to add an index to each array, since [coll] requires an index to store data correctly. Once indexes have been added to all arrays, a dump message is sent to coll alongside a “pull_from_coll” message, which imports that data into [dict] to be formatted as a .JSON file. The dump from coll sends all data out to an [unpack] to separate X and Y values, which are then grouped together again using the [thresh] object.

That list is sent to [minimum] and [maximum] for scaling both X and Y respectively. The minimum and maximum values of X and Y are then remotely sent out for pitch filtering to scale the range of the entire data set to a designated octave range. The final index of the [coll] is used to set UI objects used for selection and visualization by the [rslider] object, and [plot~]. Finally, upon making a selection, the "receive melodic_beat_number" is combined with a "get" message back to [dict] for real time sonification. As each index is output from the metronome timing in the rhythm generator, it is fed out the patch for playback.

Appendix B: Region Routing

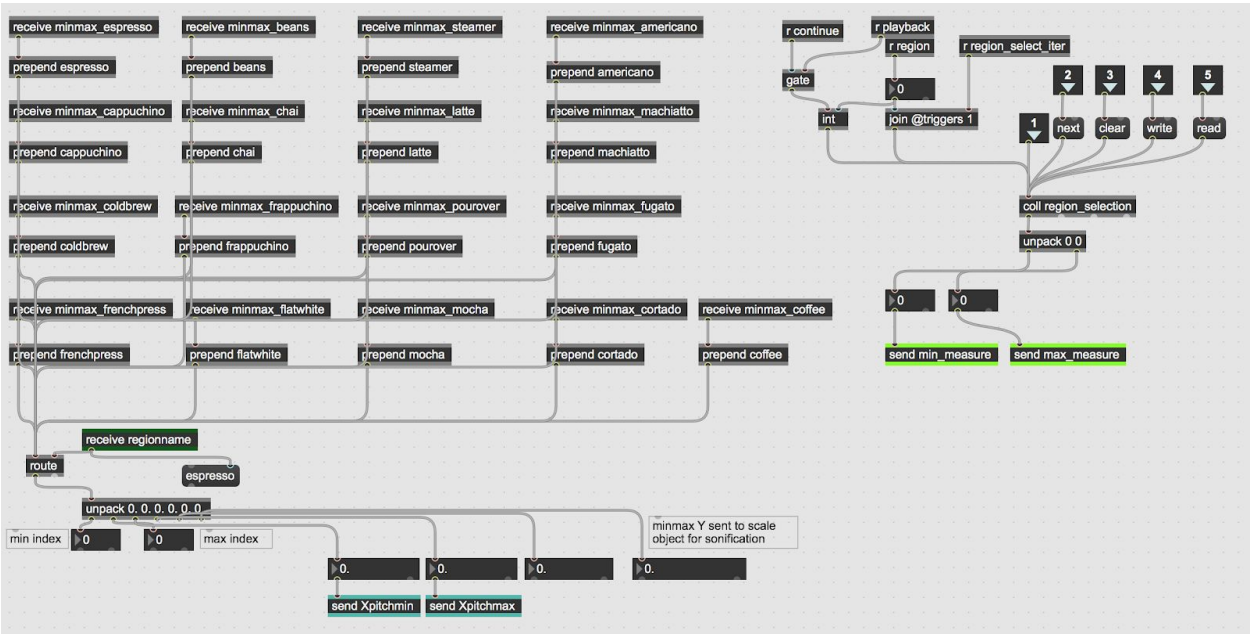


Figure 9: Region subpatch

The original purpose of this abstraction was to take a selected region and find its minimum and maximum value to create a localized sonification. This feature is not currently in use. After testing, it was decided that using the minimum and maximum values of the entire data set was a more accurate representation of data phenomena when filtered to pitches. The old code has been left in for future development, where a different kind of data may need a more localized approach.

The main feature of this abstraction is to store selections from the [rslider] object, which outputs a list of the starting and ending indexes. This list is used to set a [counter] object in the rhythm generator, which pulls indexes based on the tempo of a [metro] object. Each list stored in [coll region_selection] also contains the region number the user has selected, so that different selections of data can be recalled if the user has made more than one selection. Finally, the user can write these selections to a text document if they wish to share their selections of a data set.

Appendix C: Playback

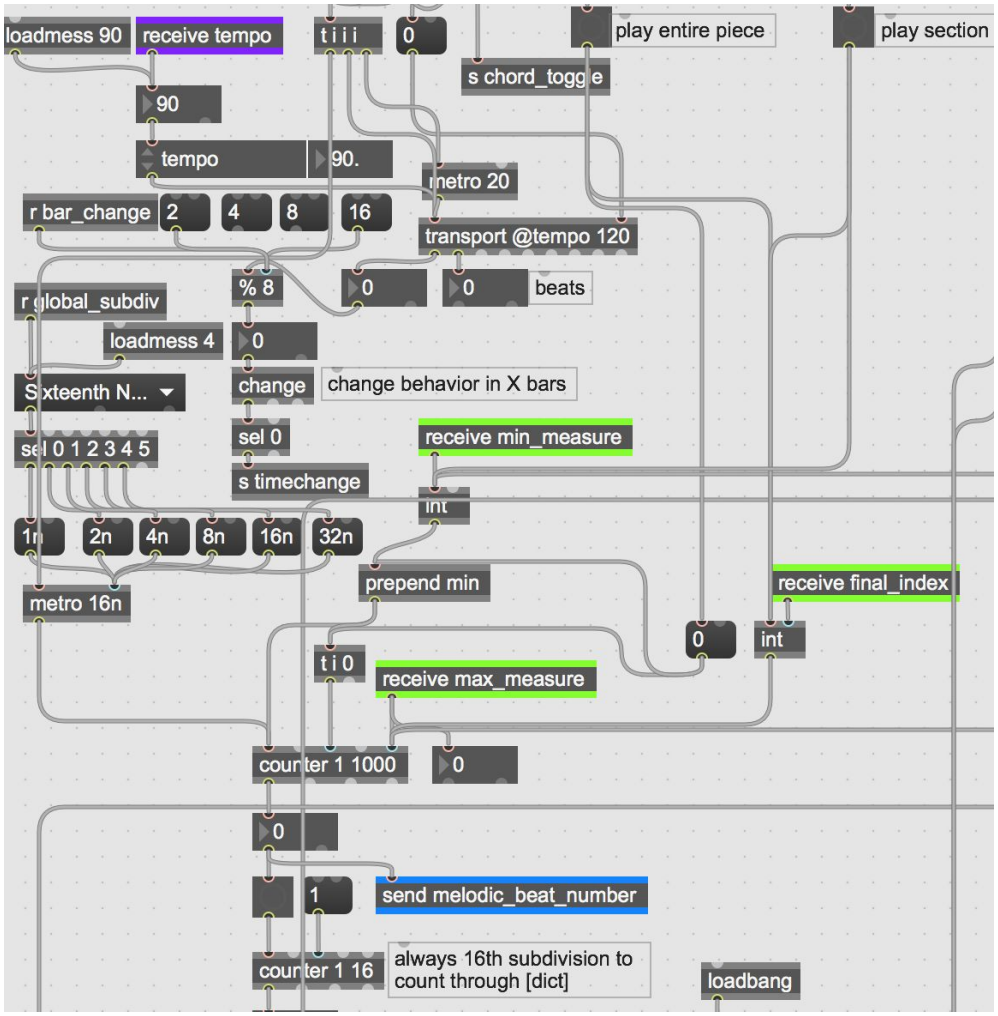


Figure 10: Playback module

This section of the patch handles indexes for real time playback. After a selection is made, the [coll] from the region routing section sends out a list, which is received by [receive min_measure] and [receive max_measure]. These are used to set a [counter] to loop at the correct place in the data set. When the uses initiates playback, a [metro] object sends bangs into [transport], which outputs beats and bars based on a tempo. The subdivision of a tempo can be set by the developer or user to change playback behavior.

As the counter is banged by another [metro] object, it outputs a value which is used to pull a single array back from [dict]. Those arrays are then sent out to the patch for pitch filtering, rhythmic changes, and finally synthesis.

Additionally, there are features to control behavior based on a number of bars that have occurred during playback. These can be sent to change parameters such as key/mode, preset behavior, or

rhythmic changes over time. It is currently not implemented in the final build, but may be useful to other developers or users in the future.

Filtering (Appendix D)

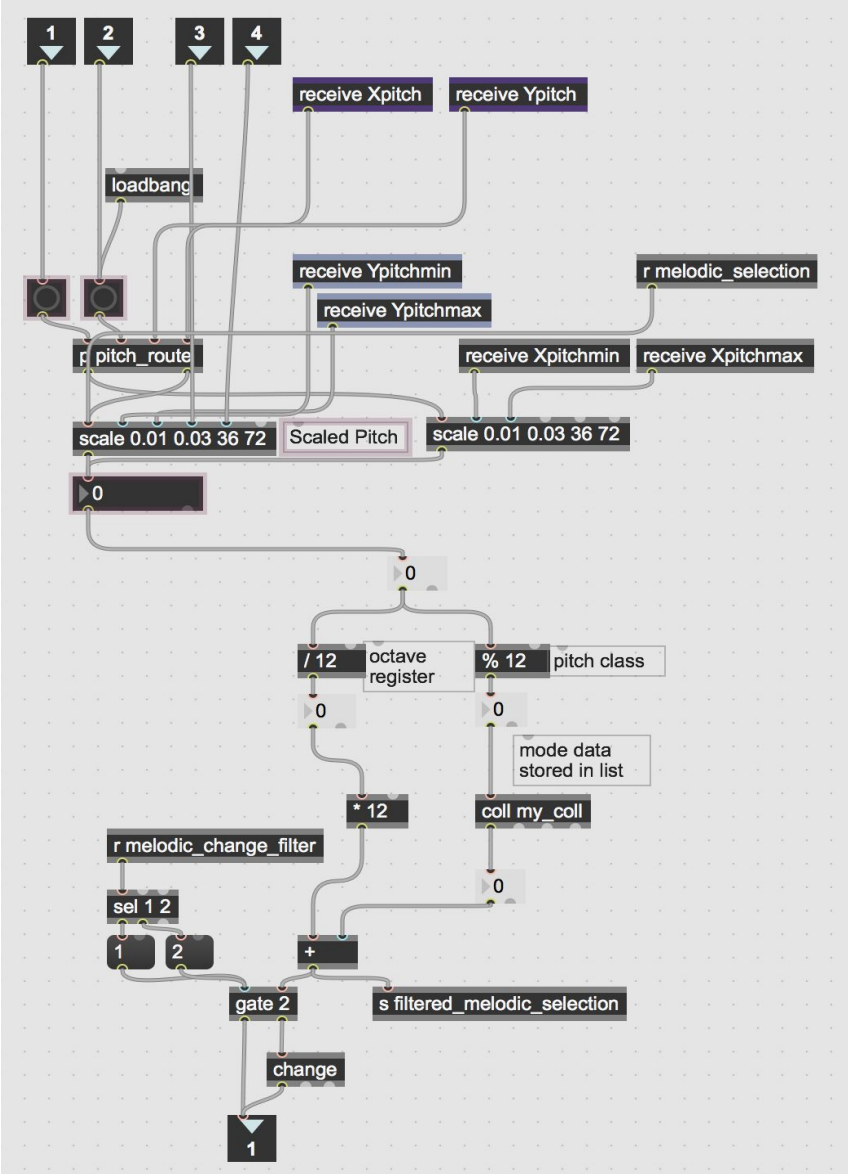


Figure 11: [p mel_filter] abstraction

As arrays of data are sent back out of [dict] from the metronome timer, they are remotely sent to this abstraction using the send and receive objects. Each array is scaled to an octave range of five octaves by default. We use Y, or "absorption rate" for spectra as our melodic voice for sonification. We use a combination of division and modulo to derive the pitch class and octave register, which are decoupled temporarily from the initial pitch value. The scaled data is then filtered again through the use of Modal Object Library objects developed by VJ Manzo. It should

be noted that other developers will need to install the EAMIR package from Max's package manager to work on the program. The [coll my_coll] stores a key and mode as a simple stepwise relationship that converts non diatonic pitches to fit. The octave register is then recombined with the filtered pitch to be sent out for audio synthesis. The gate at the bottom of the patch determines if repetitions of pitches will be output or not, which can be toggled by a user in presentation mode. By default, repetitions are avoided for playback.

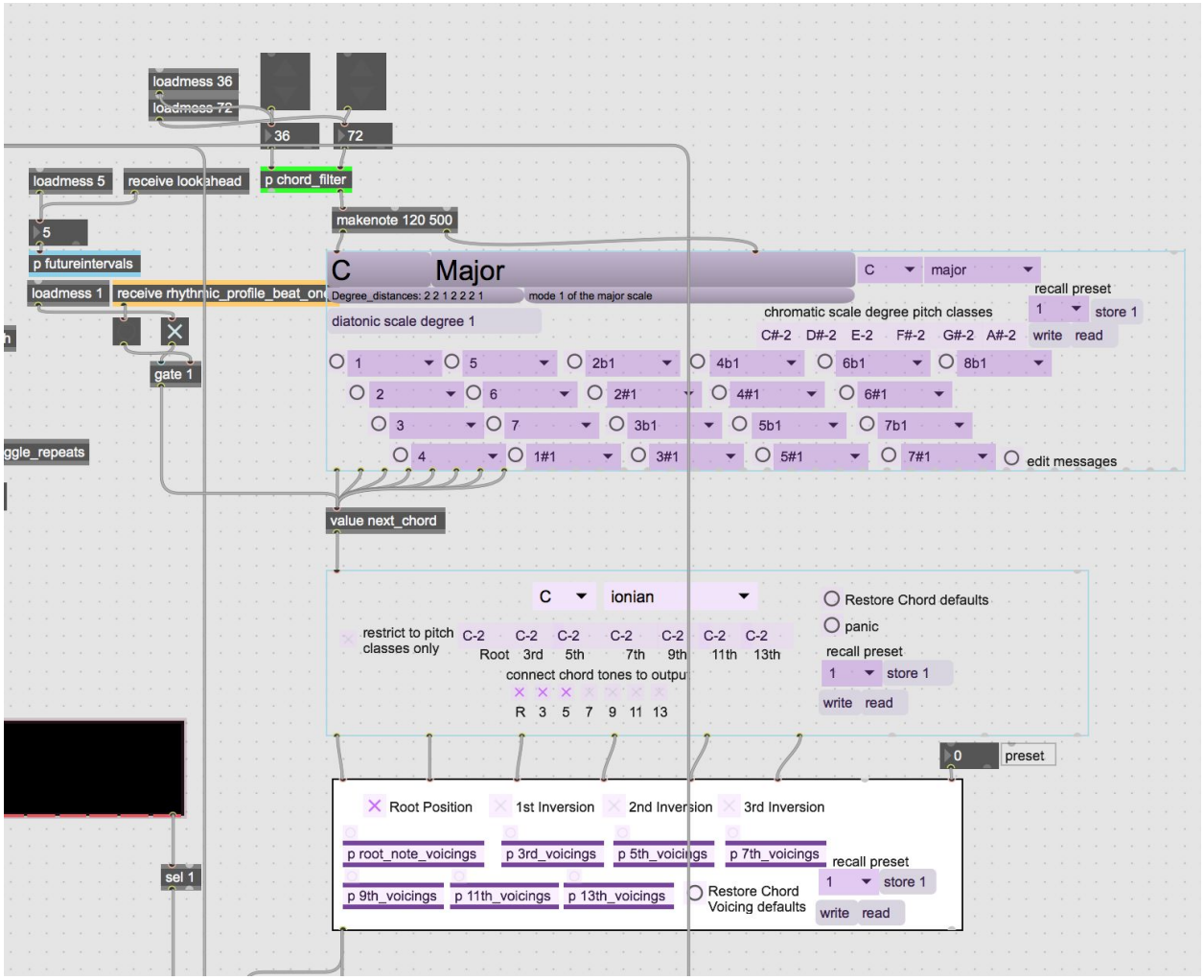


Figure 12: Chord/Harmonic Filtering

Once an X or Y value has been filtered to a diatonic pitch, it is remotely sent to a series of chordal voicing objects from the Modal Object Library. A subpatcher [p futureintervals] will pull values behind or ahead the current melodic pitch for different harmonic progressions. The pitch is sent to create a diatonic chord of the same key and mode, and can also be modified for different inversions or voicings such as a 7th chord. The resulting chord is sent to a [poly~] for audio synthesis. The root of the diatonic chord is also stripped and sent to a separate [poly~] as a bass voice.

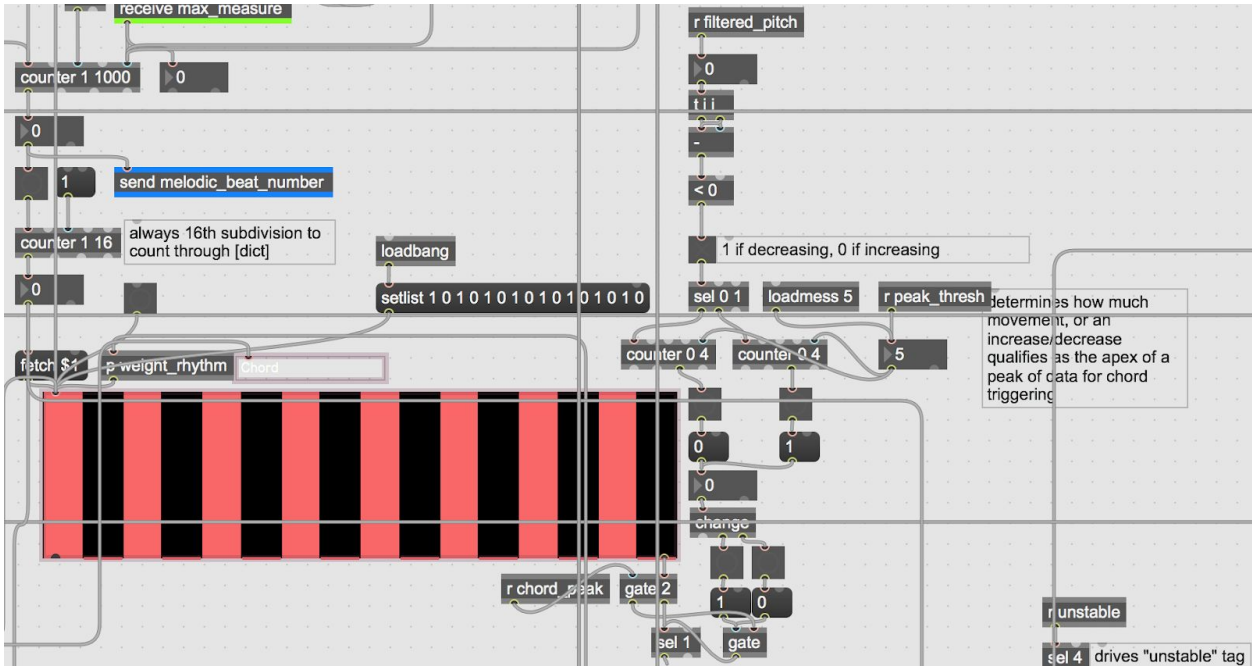


Figure 13: Peak Detection

To emphasize to a user that a peak in the data has been passed, a peak detection process was created for analyzing spectra data. This feature can be toggled on and off in the higher level patch. Peak analysis receives a filtered pitch, subtracts the pitch against itself, and looks to see if it is less than zero. The output is sent to a toggle to visualize that a data trend is decreasing or increasing. A [select] object sends a bang when it receives 1 for decrease, or 0 for increase. Each bang is sent to [counter] objects to create a threshold for when a peak should be identified. At the default setting of 5, there must be an increase/decrease of 5 pitches within the data to signify a peak. A [change] object then looks for a zero to nonzero transition, and opens a gate object to allow chords to be triggered by a rhythm mask. Chords are by default triggered in 8th notes, but can be manually changed by a user or developer depending on the data being input.

All voices are connected to “rhythm masks”, which are [multislider] objects used as a GUI to draw in different rhythms. As [transport] sends bangs, a “fetch $1” message pulls an individual slider value ber peat, by subdivision. A value of 0 or 1 is sent out of the bottom right of [multislider] to indicate if a pitch or drum sound triggers or is muted. [sel 1] looks for values of 1 and sends a bang to the corresponding filtering objects.

Synthesis and Digital Signal Processing (Appendix F)

Once data has been converted to MIDI values, it is sent to a bpatcher containing a polyphonic version of the simple BEAP oscillator patch, which uses two oscillators instead of one for more sound design possibilities.

Figure 13: Synth Bpatcher

The bpatcher allows for top level control of synthesizer parameters, including two waveform selectors, an ADSR envelope, octave control for each oscillator, and a filter for shaping.

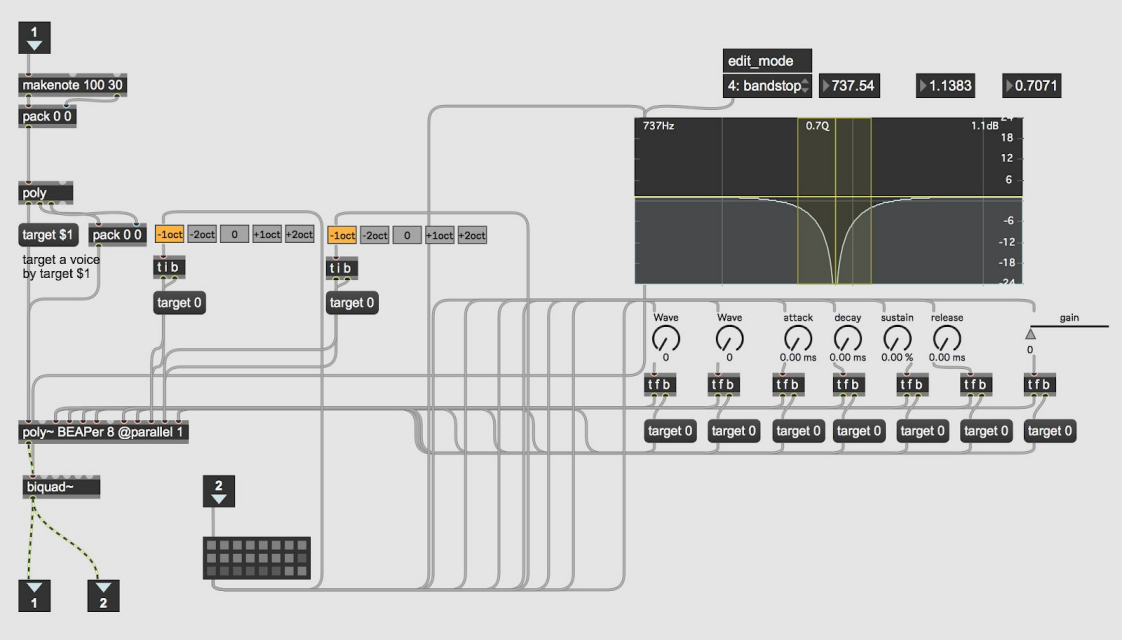


Figure 14: Overview of Bpatcher

The bpatcher contains a [poly~] patch that creates multiple instances of a BEAP oscillator. It takes MIDI in its inlet, and will target the voices of [poly~] with parameter changes like the ADSR curve. It also saves presets for callback.

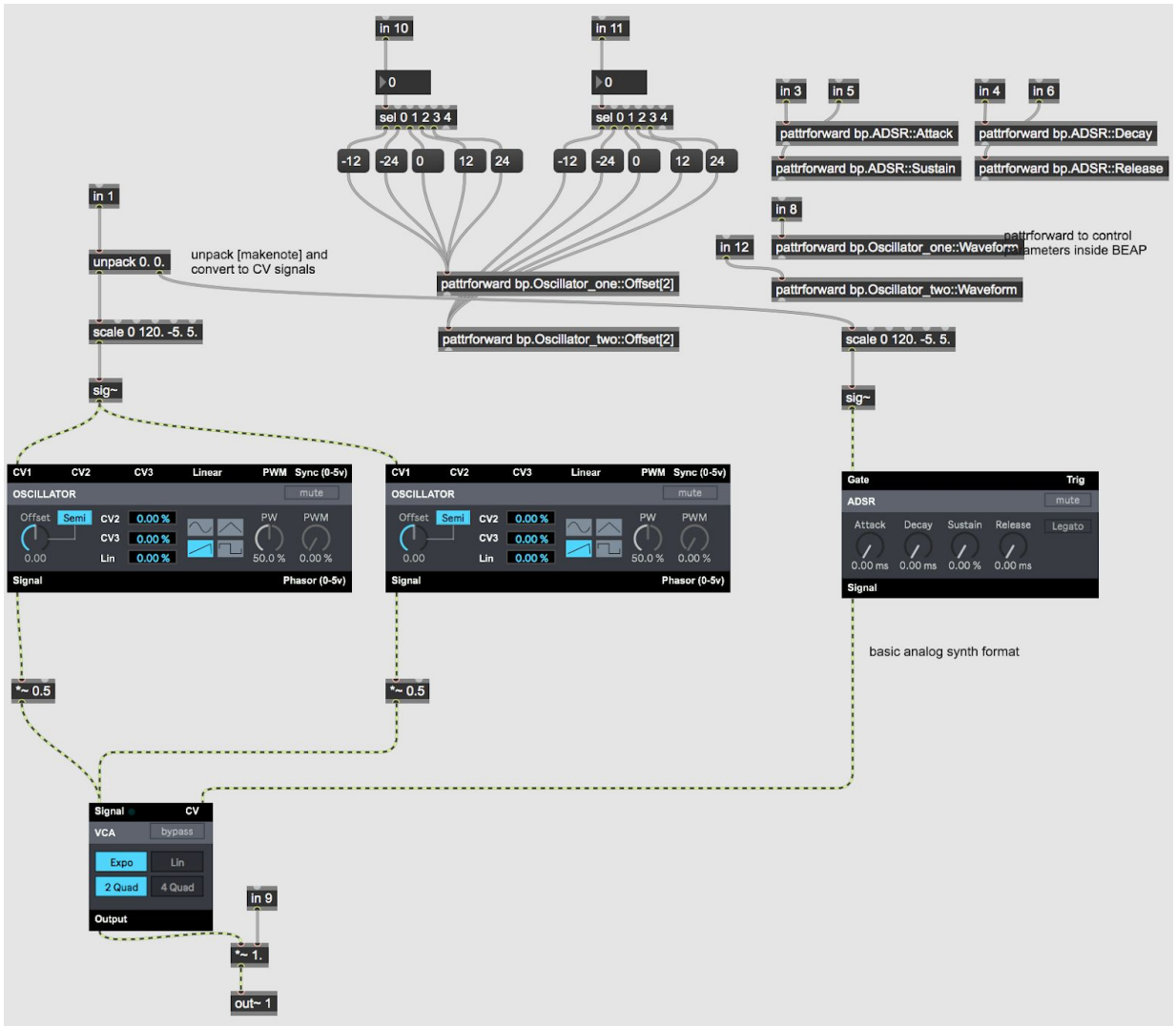


Figure 15: [poly~] overview

The [poly~] patch uses BEAP modules, which were chosen to decrease development time. BEAP patchers are used to simulate classic analog synthesizer models, and were developed to teach concepts like control voltage as an input. As the sonification system uses MIDI, a few modifications were used to implement BEAP.

[makenote] creates a MIDI note value and velocity to drive the synthesizer, but these values are unpacked upon arrival and scaled. The MIDI pitch value is scaled to the control voltage range of BEAP's oscillator, which is -5. to 5. volts. The velocity value is scaled to the same range from 0 127 to -5. and 5., but it is sent to drive a ADSR module to control note triggering. Both oscillators and the ADSR are connected to a VCA module (voltage controlled amplifier), and sent out of the [poly~] patch through its two outlets as a stereo signal.

BEAP patch parameters are hidden by default, in that they cannot be altered directly by other UI objects such as a slider or dial in Max. To work around this issue, the [pattrforward] object is used to directly access each object's parameters. Each parameter uses different values such as millisecond timings or percentage values, so each dial has been converted in the inspector options to output the appropriate value type. More parameters can be easily accessed by creating more [pattrforward] objects, which can be found by looking at View > Parameters from the Max window bar.

Presets and Tagging (Appendix G)

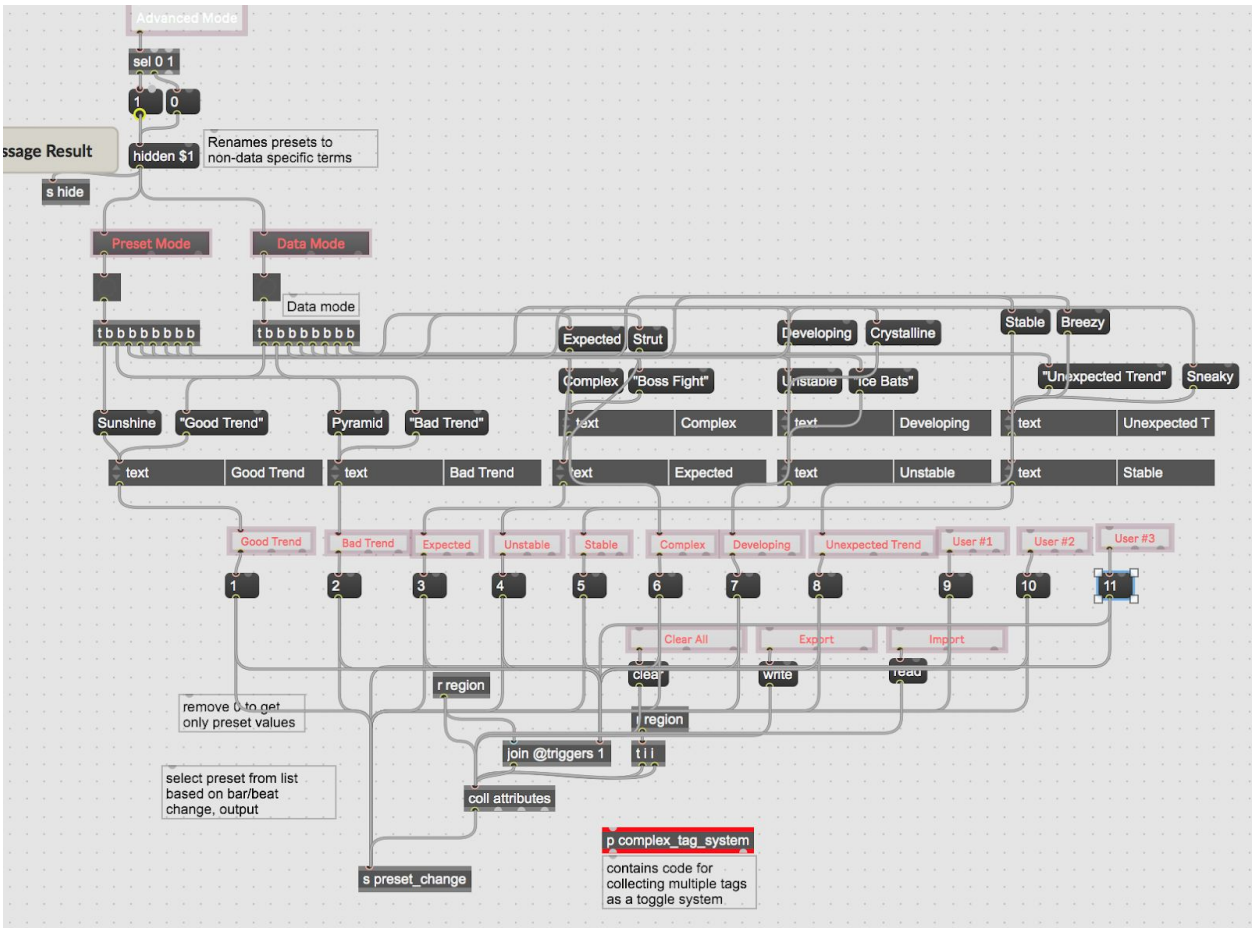


Figure 16: Tagging system

Uses a variety of [textbutton] objects to set preset changes to many musical parameters. Each button can be renamed by sending any symbol to [attrui]. This abstraction joins the region selection from the user with a chosen preset and saves it to be recalled for playback. As such, when a user swaps between their regions of data to be sonified, a tag/preset is saved onto that selection automatically.

Advanced Tag System

One removed feature was a more robust tag system that could identify when multiple tags were selected and trigger different events. This was removed due to the synthesizer's behavior when swapping between presets, which triggered loud volume spikes. Additionally, this system could be used to isolate parameters triggered by each tag, where combinations of them produce unique results.

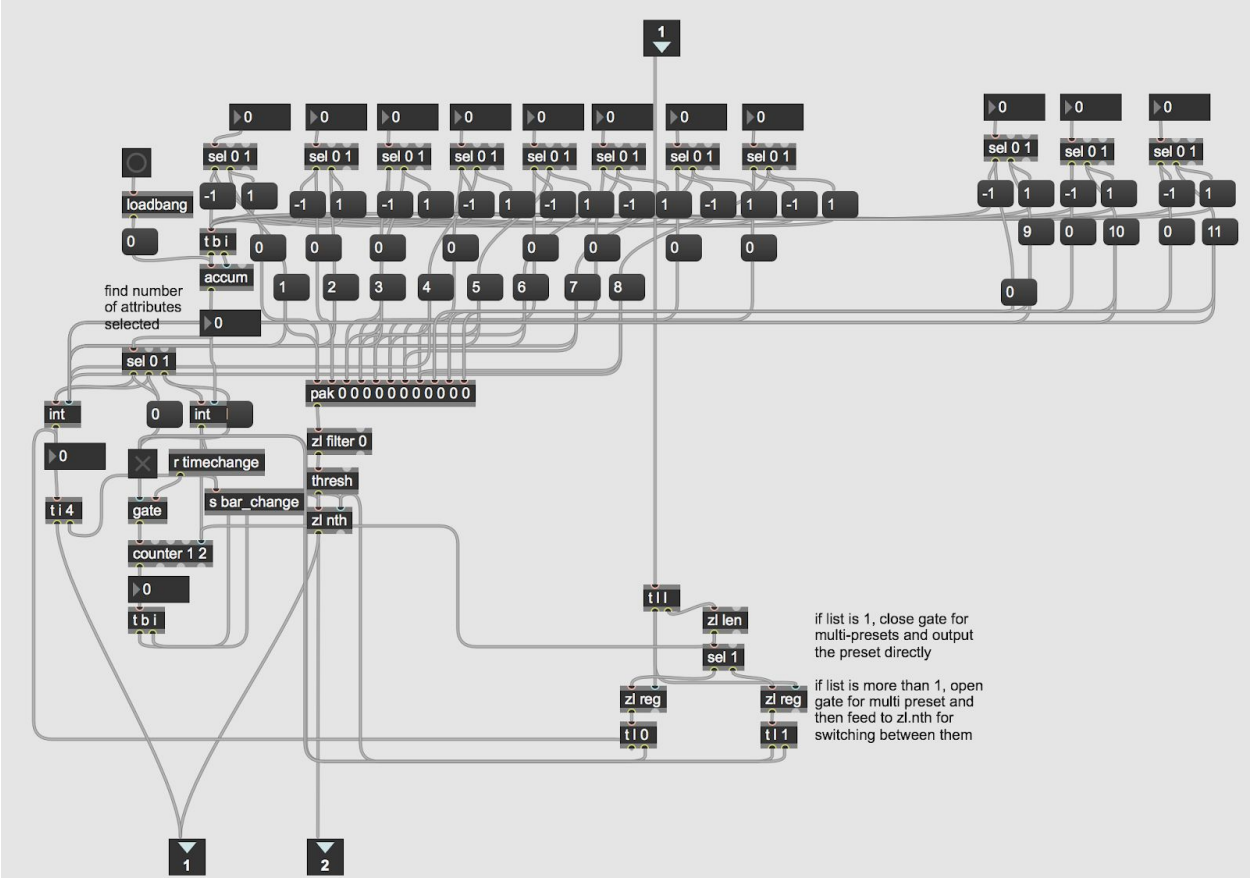


Figure 17: Advanced tagging system

In this abstraction, the [textbutton] must be set to toggle mode for input. The output of 0 or 1 is converted to -1 and 1 respectively, and sent to an [accum] object which accumulates input following a bang. If a single tag is selected, a gate object sends the preset value directly out for a parameter change. If multiple tags are selected, the gate is triggered to swap between presets over a length of time based on the [transport] object. The [pak] object collects a unique number assigned to each tag as multiple tags are selected, removes any values of zero, and then collects them into a [zl len] object to find the length of the list. This length determines how many tags have been selected, and thus, how many must be swapped around by a [transport] trigger over time, or if a single tag was selected and must be immediately sent out. The [zl reg] object stores lists and outputs them with a bang, so the list is stored twice and sent to different gates depending on its length.

Comment System (Appendix I)

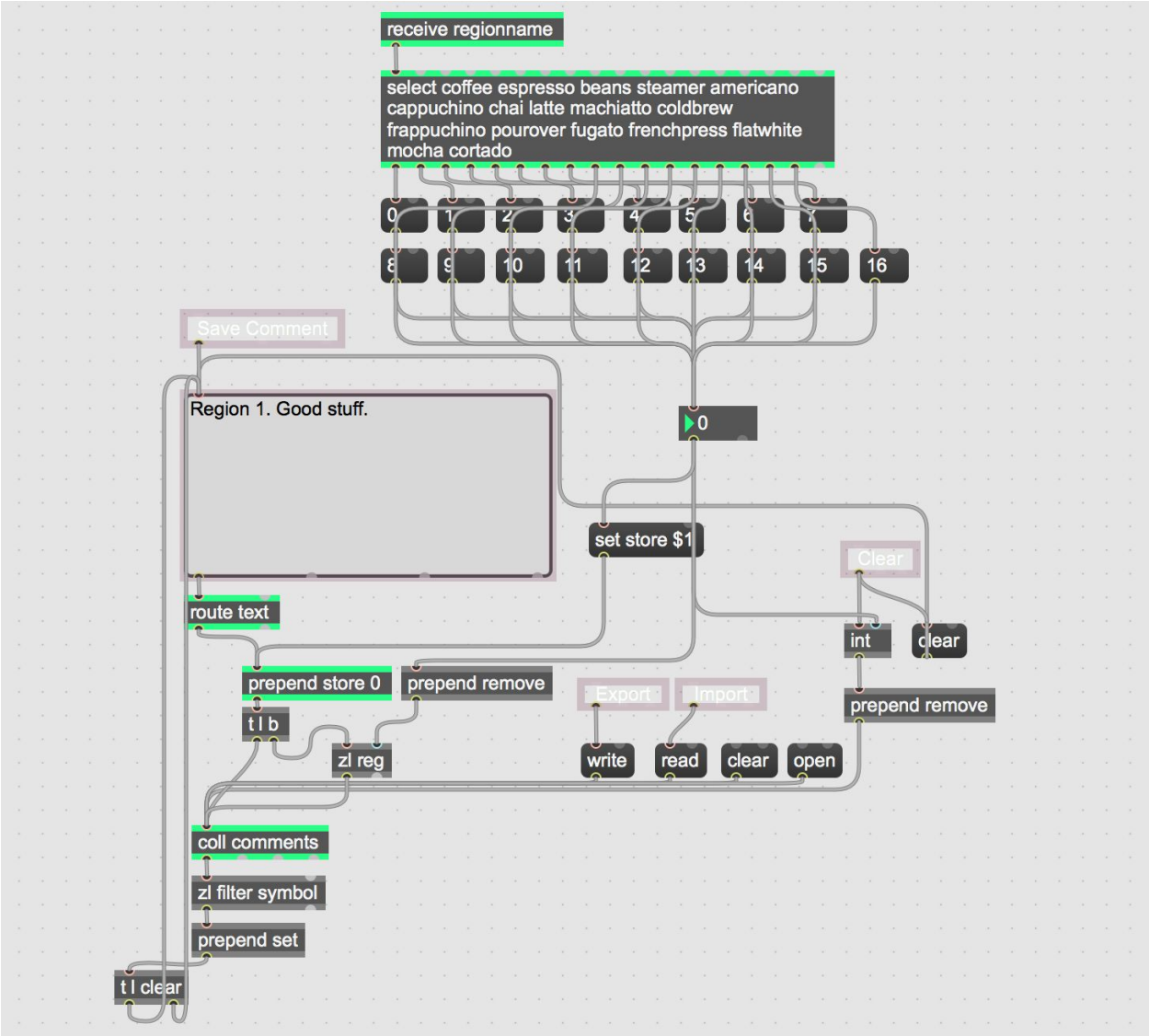


Figure 18: Comment system

Receives a selected region as an index and combines with the output of a [textedit] object to save in a [coll]. Text is entered via pressing the enter key or pressing “save comment”, which then bangs a [zl reg] that stores the list of text symbols and the index. Comments can then be imported or exported through the [coll] object.

Wavelength Mapping (Appendix J)

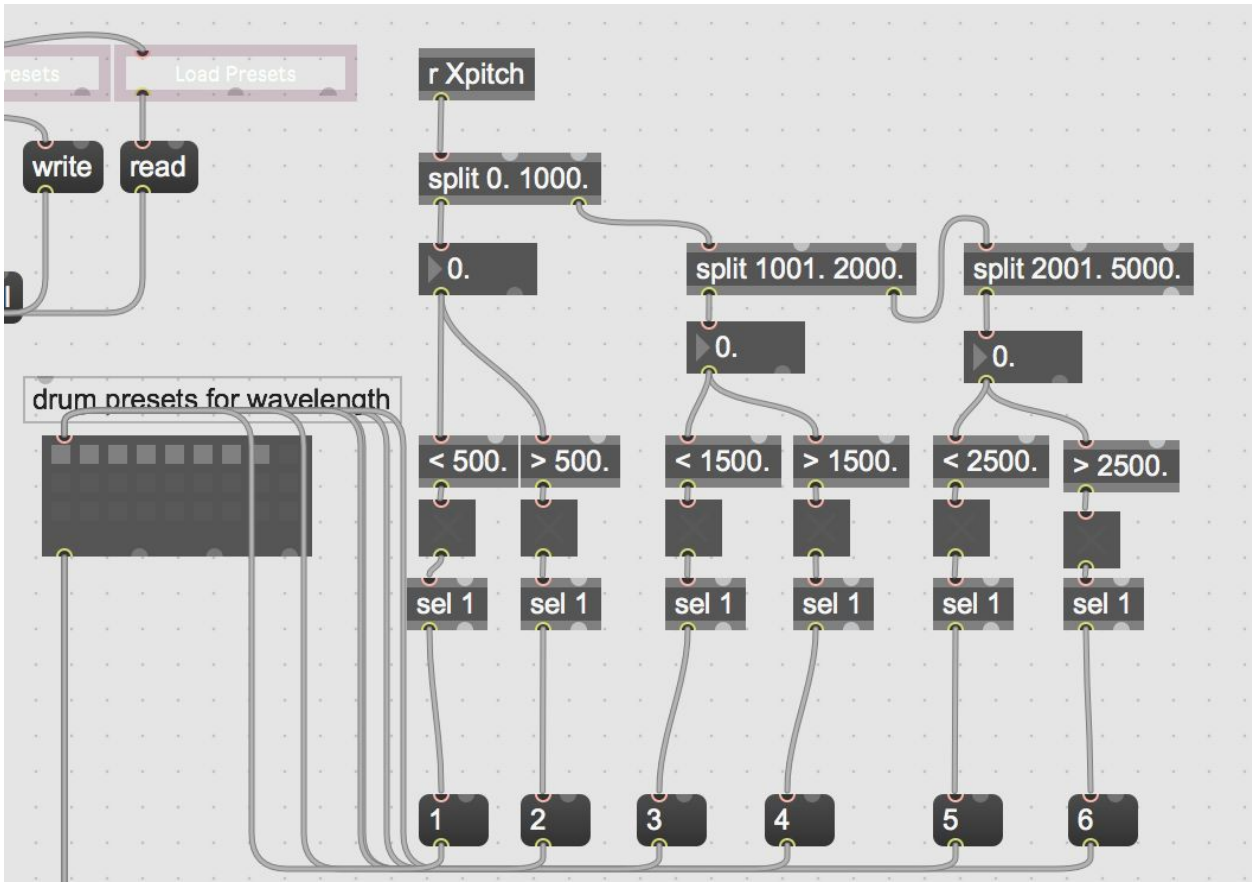


Figure 19: Wavelength mapping module

Receives X values from [dict] and looks for activity using [split] within three major regions. From 0 to 500 wavelength, no drum patterns occur. As the wavelength increases, drum patterns emphasize smaller subdivisions by triggering a preset object. These presets control each voice in the drum module, such as the kick drum and snare.

Interview Notes (Appendix K)

Short notes were taken in Trello during each interview, which were then expanded on post meeting to describe the next steps in the development process. An example of one of the meeting cards is shown below:

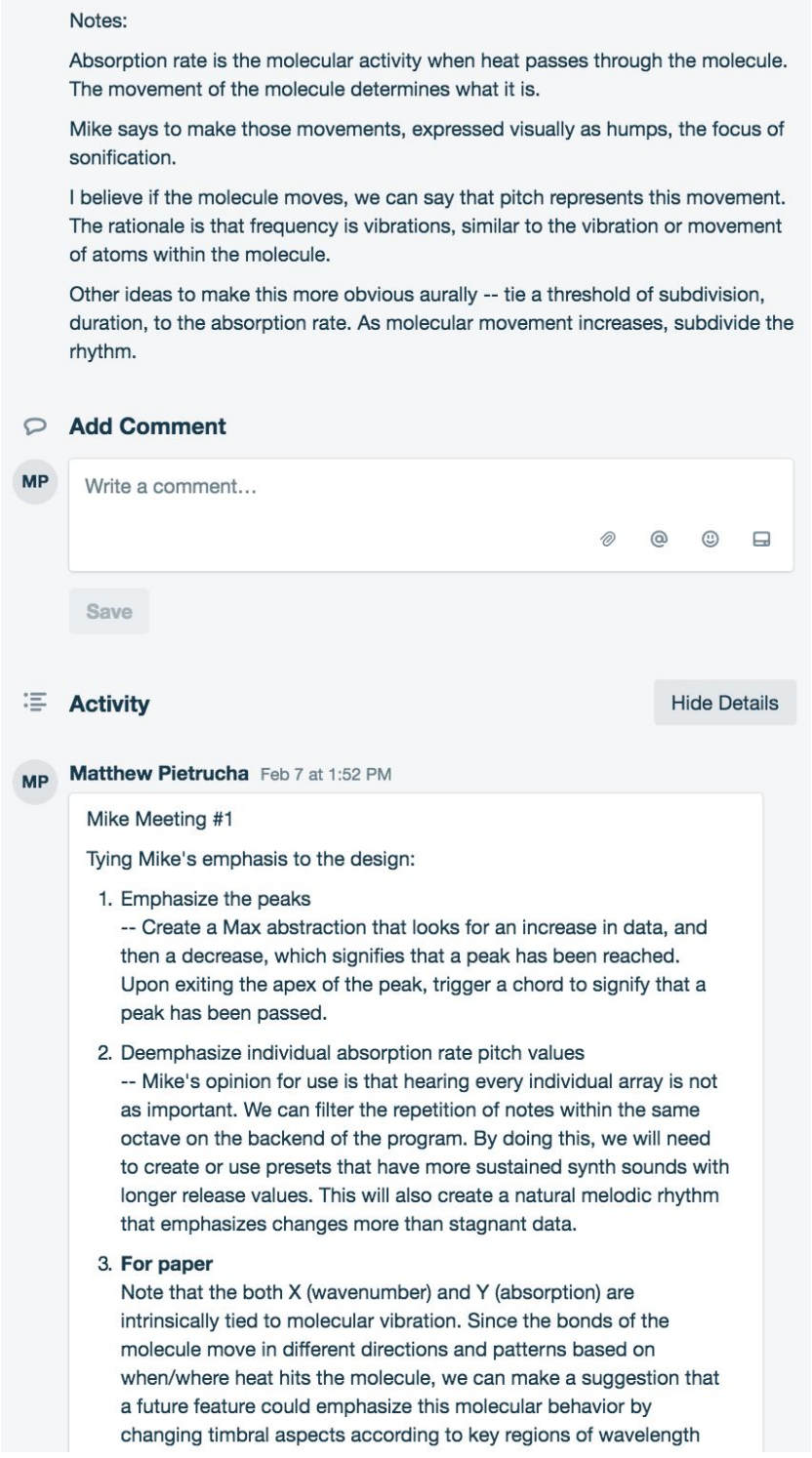
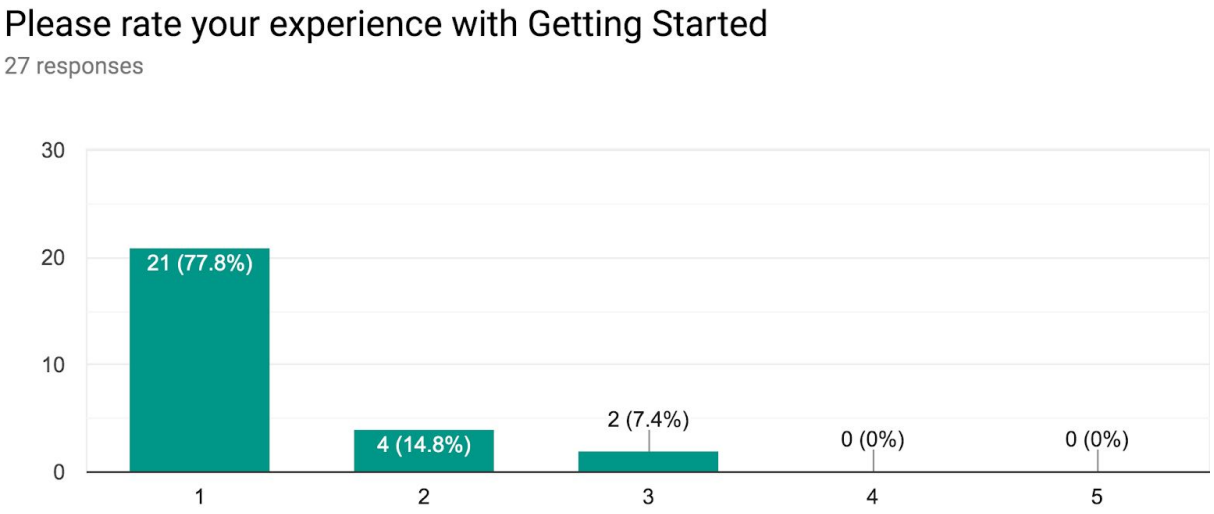


Figure 20: Interview note

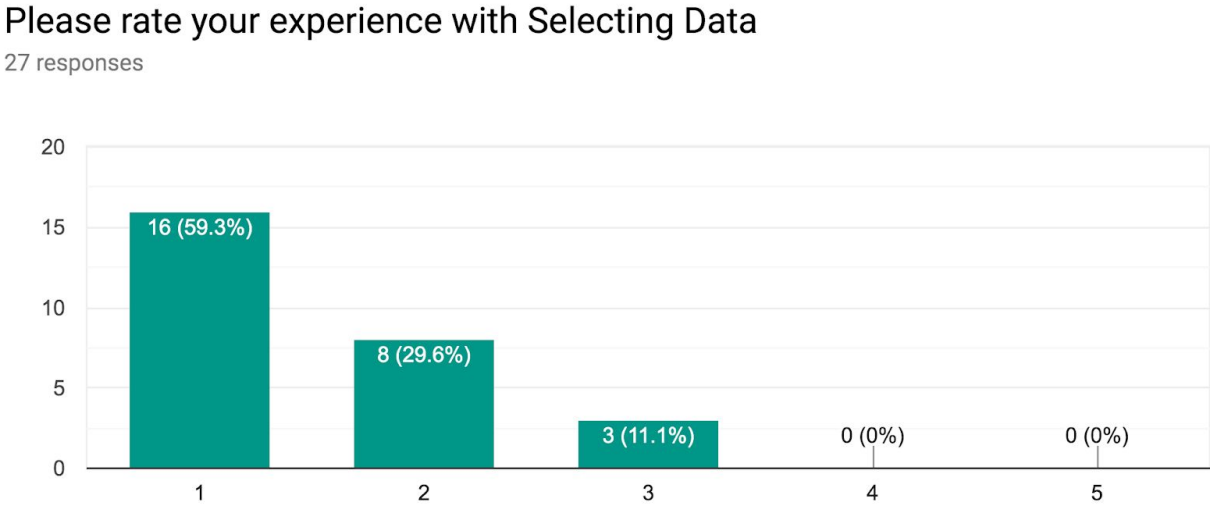
Survey Results (Appendix L)

Results of each prompt are listed below in the order they appear in the survey sequentially.

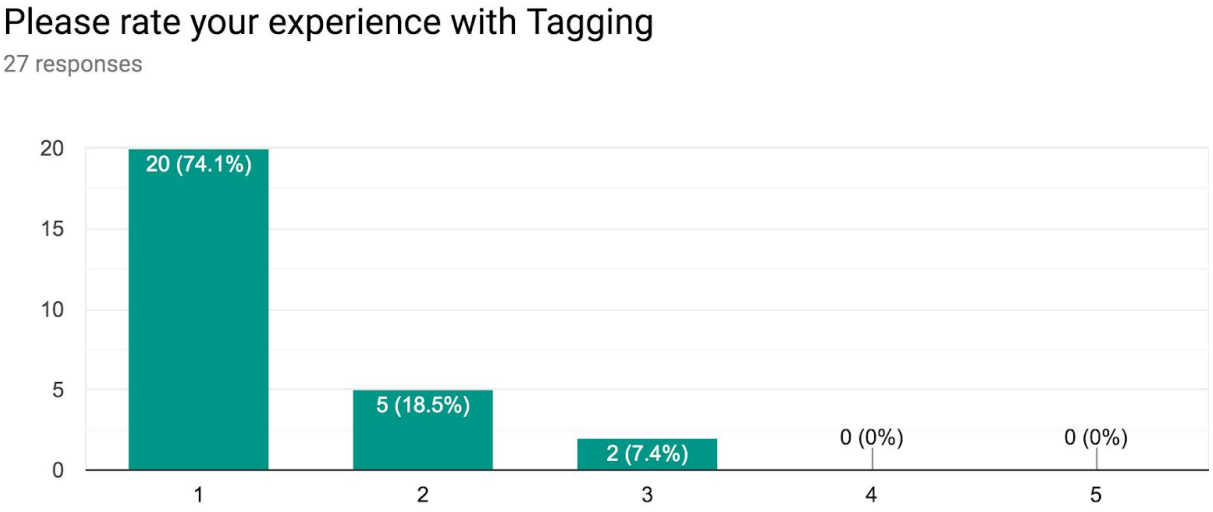
Prompt #1:



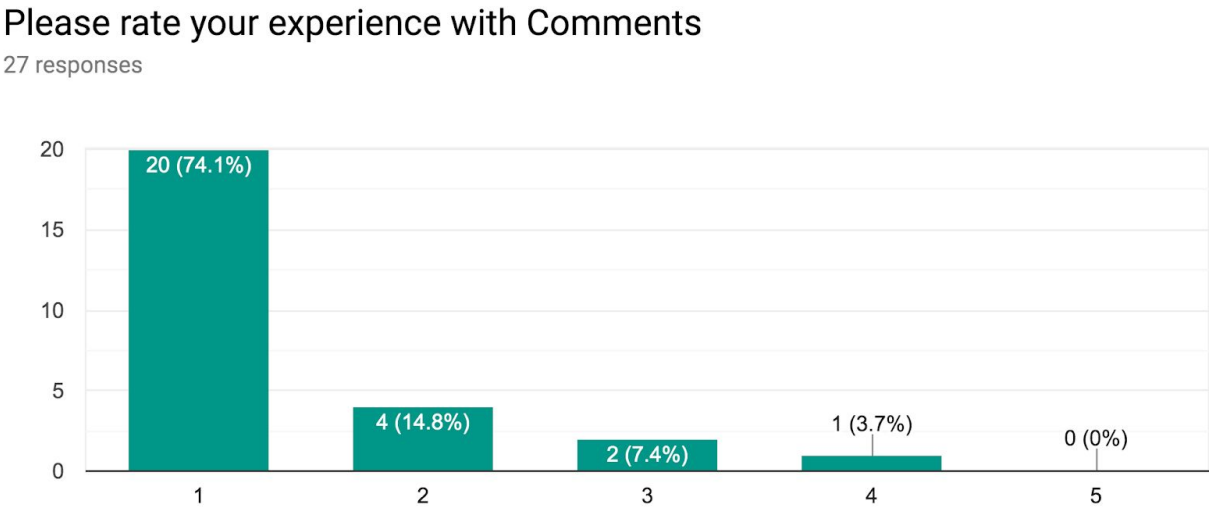
Prompt #2:



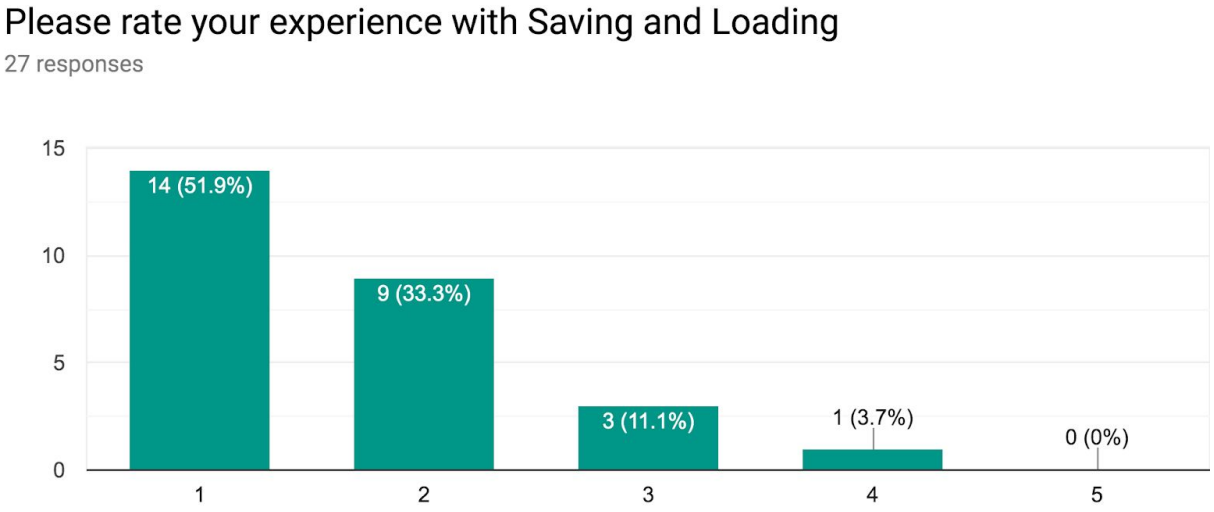
Prompt #3:



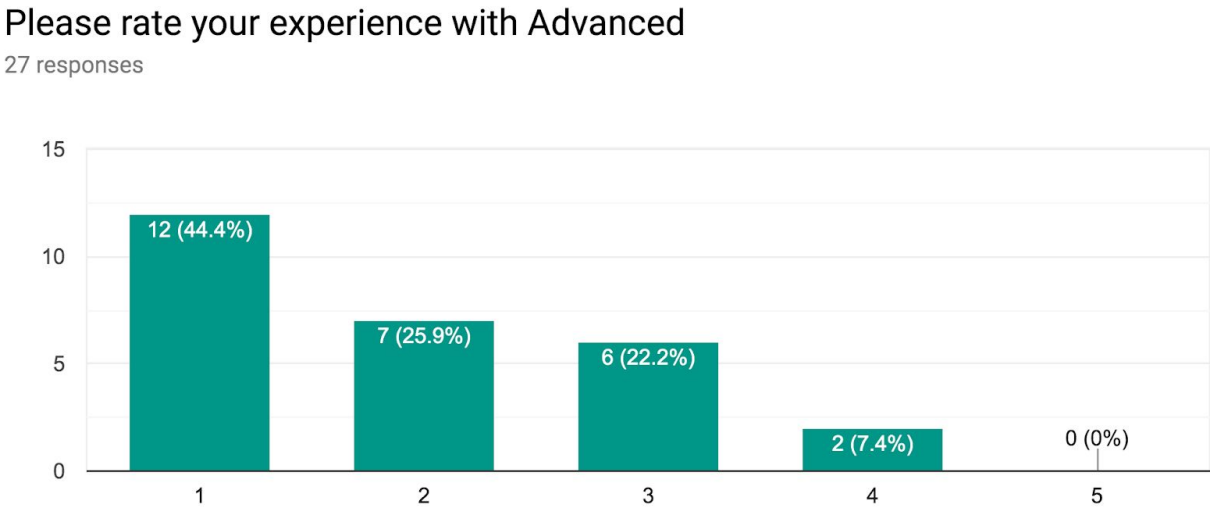
Prompt #4:



Prompt #5:



Prompt #6:



References

Barrass, S. (2011). The aesthetic turn in sonification towards a social and cultural medium. *AI & SOCIETY*, *27*(2), 177-181. doi: 10.1007/s00146-011-0335-5

Barrass, S., & Kramer, G. (1999). Using sonification. *Multimedia Systems*, *7*(1), 23-31. doi: 10.1007/s005300050108

Ballora, M., Roman, C., Pockalny, R., & Wishner, K. (2018). Sonification and Science Pedagogy: Preliminary Experiences and Assessments of Earth Science Data Presented in an Undergraduate General Education Course. *Proceedings of the 24th International Conference on Auditory Display - ICAD 2018,*213-218. doi:10.21785/icad2018.004

Brown, L. M., & Brewster, S. A. (2003). Drawing by ear: Interpreting sonified line graphs. *9th International Conference on Auditory Display (ICAD),*152-156.

Brown, A., & Sorensen, A. (2009). Integrating Creative Practice and Research in the Digital Media Arts. In Smith H. & Dean R. (Authors), *Practice-led Research, Research-led Practice in the Creative Arts* (pp. 153-165). Edinburgh: Edinburgh University Press. Retrieved from http://www.jstor.org/stable/10.3366/j.ctt1g0b594.10

Campo, A. (2007). Toward a Data Sonification Design Space Map. *The 13th International Conference on Auditory Display,*342-347. Retrieved April 23, 2019.

Childs, E. (2018). Achorripsis: A sonification of probability distributions. Retrieved April 13, 2019, from https://smartech.gatech.edu/handle/1853/51332

Cohen, J. (1993). "Kirk here:". *INTERACT 93 and CHI 93 Conference Companion on Human Factors in Computing Systems - CHI 93,*63-64. doi:10.1145/259964.260073

Cooper, I. (n.d.). BEHAVIOUR OF WAVES. Retrieved April 13, 2019, from http://www.physics.usyd.edu.au/teach_res/hsp/sp/mod31/m31_intensity.htm

Gaver, W. W., Smith, R. B., & O'Shea, T. (1992). Effective sounds in complex systems: The Arkola simulation. *Applied Ergonomics,23*(6), 429. doi:10.1016/0003-6870(92)90399-g

Hermann, T., Hunt, A., & Neuhoff, (2011). Perception, Cognition and Action in Auditory Displays, Neuhoff, J. The sonification handbook. (pp. 63-81). Logos Verlag, Berlin

Hermann, T., Hunt, A., & Neuhoff, (2011). Sonification Design and Aesthetics, Barrass, S. & Vickers, P., The sonification handbook. (pp. 145-164). Logos Verlag, Berlin

Hermann, T., Hunt, A., & Neuhoff, (2011). Interactive Sonification, Hunt, A. & Hermann, T., The sonification handbook. (pp. 273-296). Logos Verlag, Berlin

Infrared Spectroscopy: IR Energy Activates Molecular Vibrations. (2014, April 29). Retrieved April 15, 2019, from https://www.science.oregonstate.edu/~gablek/CH335/Chapter10/IR_vibrations.htm

Kramer, G., Walker, B., Bonebright, T., Cook, P., Flowers, J., Miner, N., & Neuhoff, J. (2019). Sonification Report: Status of the Field and Research Agenda. Retrieved from http://digitalcommons.unl.edu/psychfacpub/444

Manzo, V. (2019). EAMIR || eamir.org. Retrieved April 13, 2019, from http://www.eamir.org/

Martini, J., Hermann, T., Anselmetti, D., & Ritter, H. (2004). Interactive Sonification for exploring Single Molecule Properties with AFM based Force Spectroscopy. *PROCEEDINGS OF THE INT. WORKSHOP ON INTERACTIVE SONIFICATION,*1-6. Retrieved April 15, 2019.

Mynatt, E. D. (1994). Designing with auditory icons. *Conference Companion on Human Factors in Computing Systems - CHI 94,*109-119. doi:10.1145/259963.260483

Poirier-Quinot, D., Parseihian, G., & Katz, B. (2017). Comparative study on the effect of Parameter Mapping Sonification on perceived instabilities, efficiency, and accuracy in real-time interactive exploration of noisy data streams. *Displays*, *47*, 2-11. doi: 10.1016/j.displa.2016.05.001

Pereira, F., Ponte-E-Sousa, J. C., Fartaria, R. P., Bonifácio, V. D., Mata, P., Aires-De-Sousa, J., & Lobo, A. M. (2013). Sonified Infrared Spectra and Their Interpretation by Blind and Visually Impaired Students. *Journal of Chemical Education,90*(8), 1028-1031. doi:10.1021/ed4000124

Skains, R. (2018). Creative Practice as Research: Discourse on Methodology. *Media Practice And Education*, *19*(1), 82-97. doi: 10.1080/14682753.2017.1362175

Sinclair, P. AI & Soc (2012) 27: 173. Retrieved April 17, 2019, from https://doi.org/10.1007/s00146-011-0346-2

Snyder, R. (2001). *Music and Memory*. Cambridge: Bradford Books.

The Scientist and Engineer's Guide to Digital Signal Processing. (2019). Retrieved April 13, 2019, from https://dspguide.com/

Tsuchiya, T., Freeman, J., & Lerner, L. (2015). DATA-TO-MUSIC API: REAL-TIME DATA-AGNOSTIC SONIFICATION WITH MUSICAL STRUCTURE MODELS. *The 21st International Conference on Auditory Display,* 244-251. Retrieved April 15, 2019.

Vickers, P., Worrall, D., & So, R. (2017). Special Issue on Sonification. *Displays*, *47*, 1. doi: 10.1016/j.displa.2016.12.001

Walker, B. N., & Mauney, L. M. (2010). Universal Design of Auditory Graphs. *ACM Transactions on Accessible Computing,2*(3), 1-16. doi:10.1145/1714458.1714459

Xenakis, I. (2013). Pithoprakta. Retrieved April 18, 2019, from http://www.iannis-xenakis.org/xen/works/genres/work_11.html